Final

Source Code

A digital copy of this code is available at

https://github.com/daveshah1/nova

TEAM NOVA

RepRap PRO

gumstix®

Contents

1 Rover Gumstix – Python

1.1 main.py

```
1  import SocketServer
2  import arduino
3  import gps
4  import datahandler
5  import navigator
6  import time
7  from threading import Timer
8  lastCommTime = time.time()
9  deployed = False
10 class MyTCPHandler(SocketServer.BaseRequestHandler):
11
12     def handle(self):
13         global lastCommTime, deployed
14         lastCommTime = time.time()
15         self.data = self.request.recv(1024).strip()
16         print "{0} wrote:".format(self.client_address[0])
17         print self.data
18         if(self.data[3:5] == "PI"):
19             self.request.sendall("RP OK\n")
20         elif(self.data[3:5] == "CL"):
21             self.request.sendall("RP OK " + str(datahandler.latestLat) + " " +
   str(datahandler.latestLon) + "\n")
22         elif(self.data[3:5] == "MV"):
23             if(len(self.data) < 7):
24                 self.request.sendall("RP NP\n")
25             else:
26                 if(self.data[6] == "F"):
27                     deployed = True
28                     navigator.manualForward()
29                     status = True
30                 elif(self.data[6] == "L"):
31                     t = Timer(0,lambda : navigator.manualTurn(0,90)) #Run in background
32                     t.start()
33                     status = True
34                 elif(self.data[6] == "R"):
35                     t = Timer(0,lambda : navigator.manualTurn(1,90)) #Run in background
36                     t.start()
37                     status = True
38                 elif(self.data[6] == "B"):
39                     navigator.manualBackward()
40                     status = True
41                 else:
42                     status = False
43                 if(status):
44                     self.request.sendall("RP OK\n")
45                 else:
46                     self.request.sendall("RP OE\n")
47         elif(self.data[3:5] == "ST"):
48             navigator.manualStop()
49             self.request.sendall("RP OK\n")
50         elif(self.data[3:5] == "GT"):
51             splitd = self.data[6:].split(' ')
52             navigator.navigateAutomatically(float(splitd[0]),float(splitd[1]))
```

```
53              self.request.sendall("RP OK\n")
54          elif(self.data[3:5] == "FT"):
55              arduino.formatEEPROM();
56              self.request.sendall("RP OK\n")
57          elif(self.data[3:5] == "TP"):
58              #temperature, pressure = arduino.readTP()
59              if ((datahandler.latestT == -1) and (datahandler.latestP == -1)):
60                  self.request.sendall("RP OE\n")
61              else:
62                  self.request.sendall("RP OK " + str(datahandler.latestT) + " " +
    str(datahandler.latestP) + "\n")
63          elif(self.data[3:5] == "DP"):
64              realDeployed = arduino.isDeployed()
65              if(realDeployed):
66                  self.request.sendall("RP OK D\n")
67              else:
68                  self.request.sendall("RP OK N\n")
69          else:
70              self.request.sendall("RP NC")
71
72  def depCheck():
73      global deployed, lastCommTime
74      if((time.time() - lastCommTime) > 90):
75          if(arduino.isDeployed()):
76              deployed = True
77              navigator.navigateAutomatically(0,0) #Update
78          else:
79              deployed = False
80          lastCommTime = time.time()
81      else:
82          if(arduino.isDeployed() == False):
83              lastCommTime = time.time()
84
85
86
87  def main():
88      HOST, PORT = "", 9001  #Bind to 192.168.1.39, port 9001
89      server = SocketServer.TCPServer((HOST, PORT), MyTCPHandler)
90      gps.begin()
91      datahandler.begin()
92      #gpsTimer = Timer(1,gps.backgroundCheck) #Start GPS Background updater
93      #gpsTimer.start()
94
95      arduino.begin() #Open communications with Arduino
96      backgroundTimer = Timer(1,datahandler.backgroundUpdater)
97      backgroundTimer.start()
98
99      navTimer = Timer(1.5,navigator.update)
100         navTimer.start()
101
102         depCheckTimer = Timer(5,depCheck)
103         depCheckTimer.start()
104
105         print 'Starting server'
106         server.serve_forever() #Start listening for requests.
107     if __name__ == "__main__":
108         main()
```

1.2 arduino.py

```
#This file contains code related to I/O with the arduino

import serial
import time
arduinoPort = None
busy = False
STATUS_OK = 1
STATUS_OFFLINE = 2
STATUS_CMD_ERROR = 3
STATUS_PARAM_ERROR = 4
STATUS_FAULT = 5
STATUS_UNKNOWN = 6
STATUS_COMMS_ERROR = 7
class CommandResponse:
    def __init__(self,status,data):
        self.status = status
        self.data = data


def sendRequest(command,data):
    global arduinoPort, busy
    try:
        start_time = time.time()
        while(busy):
            if((time.time() - start_time()) > 2):
                return CommandResponse(STATUS_COMMS_ERROR,"")
        busy = True
        arduinoPort.write("HI " + command + " " + data + "\n")
        response = arduinoPort.readline()
        busy = False
        if(len(response) < 5):
            return CommandResponse(STATUS_COMMS_ERROR)
        if(len(response) < 7):
            data = ""
        else:
            data = response[6:]

        if(response[3:5] == "OK"):
            return CommandResponse(STATUS_OK,data)
        elif(response[3:5] == "CE"):
            return CommandResponse(STATUS_COMMS_ERROR,data)
        elif(response[3:5] == "NP"):
            return CommandResponse(STATUS_PARAM_ERROR,data)
        elif(response[3:5] == "NC"):
            return CommandResponse(STATUS_CMD_ERROR,data)
        elif(response[3:5] == "OE"):
            return CommandResponse(STATUS_FAULT,data)
        else:
            return CommandResponse(STATUS_COMMS_ERROR,data)
    except:
        busy = False
        return CommandResponse(STATUS_COMMS_ERROR,"")

def testComms():
    response = sendRequest("PI","")
```

```
56      if(response.status == STATUS_OK):
57          return True
58      else:
59          return False
60
61  def stop():
62      try:
63          sendRequest("MT","S S")
64      except:
65          pass
66      try:
67          arduinoPort.close()
68      except:
69          pass
70
71  def begin():
72      global arduinoPort
73      #Arduino is on UART1
74      arduinoPort = serial.Serial("/dev/ttyS0",9600,timeout=3)
75      #Make sure Arduino and port is ready
76      time.sleep(0.5)
77      #Ping Arduino
78      if(testComms()):
79          return True
80      else:
81          stop()
82          return False
83
84  def motorCtl(left,right):
85      #Try up to 3 times
86      for i in range(1,3):
87          response = sendRequest("MT",left + " " + right)
88          if(response.status == STATUS_OK):
89              return True
90      return False
91  #Returns a tuple in form (temperature, pressure) or (-1,-1) if unsuccessful
92  def readTP():
93      #Try up to 3 times
94      for i in range(1,3):
95          response = sendRequest("TP","")
96          if(response.status == STATUS_OK):
97              splitstr = response.data.split(" ")
98              if(len(splitstr) < 2):
99                  continue
100                 return ( float(splitstr[0]) / 10, float(splitstr[1]) )
101         return (-1,-1)
102
103     def isDeployed():
104         for i in range(1,3):
105             response = sendRequest("TP","")
106             if(response.status == STATUS_OK):
107                 if(response.data == "D"):
108                     return True
109                 else:
110                     return False
111         return False
112
```

```
113     def formatEEPROM():
114         for i in range(1,3):
115             response = sendRequest("FT","")
116             if(response.status == STATUS_OK):
117                 return True
118         return False
```

1.3 cam.py

```
1   import commands, os, time, shutil
2   while True:
3         status, output = commands.getstatusoutput("mplayer tv:// -vo png:z=7 -ss 1 -frames 1 -loop 1 -tv driver=v4l2:device=/dev/video0")
4         print output
5         shutil.copy("00000001.png",os.environ['DIR'] + "/" + time.strftime("%H:%M:%S") + "-cap.png")
6         os.rename("00000001.png","/var/www/image.png")
7         time.sleep(0.1)
```

1.4 datahandler.py

```
1   import time
2   import arduino
3   import gps
4   import traceback
5   import subprocess
6   import os
7   import commands
8   from pynmea import nmea
9
10  latestT = 0.0
11  latestP = 0.0
12  latestLat = 0.0
13  latestLong = 0.0
14  latestAlt = 0.0
15  GPSavailable = False
16  DateTimeSet = False
17  def putLogLine(s):
18      with open(os.environ['DIR'] + '/log.csv', 'a') as f:
19          f.write(s + "\n")
20
21  def begin():
22      putLogLine("Time, Temp, Pressure, Lat, Long, Alt, GPS Status")
23
24  def scheduledUpdate():
25      global latestT, latestP, latestLat, latestLon, latestAlt, GPSavailable, DateTimeSet
26      latestT, latestP = arduino.readTP()
27      sentence = gps.getSentence("$GPGGA",3)
28      if(sentence == ""): #getSentence returns blank in case of error
29          GPSavailable = False
30      else:
31          gpgga = nmea.GPGGA()
32          gpgga.parse(sentence)
33          if(gpgga.gps_qual == "0"):
34              GPSAvailable = False
35          else:
36              lat = gpgga.latitude
37              lon = gpgga.longitude
38              realLat = float(lat[:2]) + (float(lat[2:])/60)
```

```
39          realLon = float(lon[:3]) + (float(lon[3:])/60)
40          if("S" in gpgga.lat_direction):
41              realLat = -realLat
42          if("W" in gpgga.lon_direction):
43              realLon = -realLon
44          latestLat = realLat
45          latestLon = realLon
46          latestAlt = gpgga.antenna_altitude
47          if(DateTimeSet == False):
48              try:
49                  dateTimeInfo = gps.getSentence("$GPRMC",1.5)
50                  if(dateTimeInfo != ""):
51                      gprmc = nmea.GPRMC()
52                      gprmc.parse(dateTimeInfo)
53                      if(len(gprmc.timestamp)>0):
54                          print gprmc.timestamp
55                          hours = gprmc.timestamp[:2]
56                          minutes = gprmc.timestamp[2:4]
57                          seconds = gprmc.timestamp[4:6]
58                          year = gprmc.datestamp[4:6]
59                          month = gprmc.datestamp[2:4]
60                          day = gprmc.datestamp[:2]
61                          timestring = year + "-" + month + "-" + day
62                          timestring += " "  + hours + ":" + minutes + ":" + seconds
63                          print timestring
64                          #Sync system time to GPS Time
65                          status, output = commands.getstatusoutput("date -s \"" + timestring
    + "\"")
66                          print "Date:: " + output
67                          DateTimeSet = True
68              except:
69                  print "Oops! in datetime sync code"
70                  print traceback.format_exc()
71      csvLine = time.strftime("%H:%M:%S") + ","
72      csvLine += str(latestT) + "," + str(latestP) + ","
73      csvLine += str(latestLat) + "," + str(latestLong) + "," + str(latestAlt)
74      csvLine += "," + str(GPSavailable)
75      putLogLine(csvLine)
76  def backgroundUpdater():
77      while(True):
78          startTime = time.time()
79          try:
80              scheduledUpdate()
81          except:
82              print "Oops! in scheduledUpdate"
83              print traceback.format_exc()
84          while((time.time() - startTime) < 1):
85              pass
```

1.5 gps.py

```
1   import serial
2   import time, os
3   import traceback
4   gpsPort = None
5   currentSentence = ""
6   debug = False
7   def begin():
```

```
    global gpsPort
    #Open serial port
    gpsPort = serial.Serial("/dev/ttyS1",9600,timeout=3)

def getSentence(code, timeout):
    global gpsPort, debug, currentSentence
    startTime = time.time()
    while((time.time() - startTime) < timeout):
        try:
            s = gpsPort.readline()
            if(len(s) > 10):
                if(s[:6] == code): #Only interested in position updates for now
                    if(debug):
                        print "Got " + code + ": " + s
                    with open(os.environ['DIR'] + '/gps.log', 'a') as f:
                        f.write("Recieved Sentence: " + s + "\n")
                    return s
        except KeyboardInterrupt:
            exit(0)
        except:
            try:
                #Log exception for failure analysis
                with open(os.environ['DIR'] + '/gps.log', 'a') as f:
                    f.write("+++++ EXCEPTION +++++\n")
                    f.write(traceback.format_exc() + "\n")
                if(debug):
                    print traceback.format_exc()
            except:
                pass #Just incase there is an exception in the exception handler...
            return ""
    return ""


#begin()

def main():
    global debug
    debug = True
    begin()
    while(True):
        getSentence("$GPGGA",3)
if __name__ == "__main__":
    main()
```

1.6 navigator.py

```
import datahandler
import arduino
import time
from math import radians, cos, sin, asin, sqrt, degrees, atan2

"""
AUTOMATIC NAVIGATION ROUTINE
General Notes:
 - For all turns, 0 = left, 1 = right
 - update() must be run at regular intervals
 - To auto-cal the system, set it to navigate around an area.
   After approx 10 turns calibration will be valid
```

```
 - The system depends on datahandler having valid GPS data.
 - Without valid GPS data, behaviour will be unpredictable.
"""

"""
Coefficients used to determine how long to turn
Number of milliseconds to turn 90 deg
"""
turnCoefficientLeftDefault = 1000
turnCoefficientRightDefault = 1000

#These values will be loaded from file and automatically tuned
turnCoefficientLeft = turnCoefficientLeftDefault
turnCoefficientRight = turnCoefficientRightDefault

#Time and direction of last turn
#Used for autocal
timeLastTurn = 0
directionLastTurn = 0
angleLastTurn = 0

#Current estimated rover direction (as a bearing from North)
estimatedRoverDirection = 0

"""
How well the rover knows what direction it is facing
0: Not known at all
1: Known accurately, not moved since calculation
2: Guess, moved since calculation
"""
directionValidityStatus = 0

#GPS coordinates at last reference point
refLat = 0
refLon = 0

#Target points to navigate to
targetLat = 0
targetLon = 0

#Are we navigating?
automaticNavigation = False
lastTurnAutomatic = False
def approx_Equal(x, y, tolerance=0.001):
    return abs(x-y) <= 0.5 * tolerance * (x + y)


def distBearing(lon1, lat1, lon2, lat2):
    """
    Calculate the great circle distance between two points
    on the earth (specified in decimal degrees)
    """
    # convert decimal degrees to radians
    lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])

    # haversine formula
    dlon = lon2 - lon1
```

```
70      dlat = lat2 - lat1
71      a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
72      c = 2 * asin(sqrt(a))
73
74      # 6367 km is the radius of the Earth
75      km = 6367 * c
76
77      Bearing = degrees(atan2(cos(lat1)*sin(lat2)-sin(lat1)*cos(lat2)*cos(lon2-lon1),
    sin(lon2-lon1)*cos(lat2)))
78      return km*1000, ((Bearing+360)%360)
79
80  def begin():
81      global turnCoefficientLeft, turnCoefficientRight
82      try:
83          with open('/home/root/turncal', 'r') as f:
84              line = f.readline()
85              splitline = ",".split(line)
86              turnCoefficientLeft = float(splitline[0])
87              turnCoefficientRight = float(splitline[1])
88      except:
89          pass #File not opened, use default values
90
91
92
93  """
94  Call regularly to continue background navigation tasks
95  """
96  def update():
97      global turnCoefficientLeft, turnCoefficientRight
98      global refLat, refLon, targetLat, targetLon
99      global automaticNavigation, lastTurnAutomatic, directionLastTurn,
    estimatedRoverDirection, angleLastTurn
100         global directionValidityStatus
101         if(automaticNavigation):
102             currentLat = datahandler.latestLat
103             currentLon = datahandler.latestLon
104             if(approx_Equal(currentLat,targetLat,0.00005) and
    approx_Equal(currentLon,targetLon,0.00005)):
105                 automaticNavigation = False
106                 arduino.motorCtl("S","S")
107                 return
108             distance, bearing = distBearing(currentLon,currentLat,targetLon,targetLat)
109             distanceTravelled, currentBearing =
    distBearing(refLon,refLat,currentLon,currentLat)
110             if(distanceTravelled > 7):
111                 arduino.motorCtl("S","S")
112                 if(directionLastTurn == 0):
113                     delta = currentBearing - estimatedRoverDirection
114                 else:
115                     delta = estimatedRoverDirection - currentBearing
116                 estimatedRoverDirection = bearing
117                 if((directionValidityStatus >= 1) and lastTurnAutomatic ):
118                     lastTurnCoefficient = (abs(delta) / 90) / timeLastTurn
119                     if(directionLastTurn == 0):
120                         turnCoefficientLeft = (turnCoefficientLeft * 10 + lastTurnCoefficient) /
    11
121                     else:
```

```
122                 turnCoefficientRight = (turnCoefficientRight * 10 + lastTurnCoefficient) /
    11
123             with open('/home/root/turncal', 'w') as f:
124                 f.write(str(turnCoefficientLeft) + "," + str(turnCoefficientRight) + "\n")
125         directionValidityStatus = 1
126         amountToTurn = currentBearing - bearing
127         if(abs(amountToTurn) > 10):
128             directionValidityStatus = 2
129             lastTurnAutomatic = True
130             if(amountToTurn > 0):
131                 timeToTurn = (amountToTurn / 90) * turnCoefficientLeft
132                 angleLastTurn = amountToTurn
133                 directionLastTurn = 0
134                 estimatedRoverDirection -= amountToTurn
135                 estimatedRoverDirection = (estimatedRoverDirection + 360) % 360
136                 arduino.motorCtl("B","F")
137                 time.sleep(timeToTurn / 1000)
138                 arduino.motorCtl("S","S")
139             else:
140                 timeToTurn = (abs(amountToTurn) / 90) * turnCoefficientRight
141                 angleLastTurn = amountToTurn
142                 directionLastTurn = 1
143                 estimatedRoverDirection += amountToTurn
144                 estimatedRoverDirection = (estimatedRoverDirection + 360) % 360
145                 arduino.motorCtl("F","B")
146                 time.sleep(timeToTurn / 1000)
147                 arduino.motorCtl("S","S")
148         refLon = currentLon
149         refLat = currentLat
150     arduino.motorCtl("F","F")
151
152 def manualTurn(direction, angle):
153     global automaticNavigation,estimatedRoverDirection,lastTurnAutomatic
154     automaticNavigation = False
155     lastTurnAutomatic = False
156     if(direction == 0):
157         timeToTurn = (abs(angle) / 90) * turnCoefficientLeft
158         estimatedRoverDirection -= angle
159         estimatedRoverDirection = (estimatedRoverDirection + 360) % 360
160         arduino.motorCtl("B","F")
161         time.sleep(timeToTurn / 1000)
162         arduino.motorCtl("S","S")
163     else:
164         timeToTurn = (abs(angle) / 90) * turnCoefficientLeft
165         estimatedRoverDirection += angle
166         estimatedRoverDirection = (estimatedRoverDirection + 360) % 360
167         arduino.motorCtl("F","B")
168         time.sleep(timeToTurn / 1000)
169         arduino.motorCtl("S","S")
170
171 def manualStop():
172     global automaticNavigation
173     automaticNavigation = False
174     arduino.motorCtl("S","S")
175
176 def manualForward():
177     global automaticNavigation
```

```
178         automaticNavigation = False
179         arduino.motorCtl("F","F")
180 
181     def manualBackward():
182         global automaticNavigation
183         automaticNavigation = False
184         arduino.motorCtl("B","B")
185 
186     def navigateAutomatically(lat,lon):
187         global targetLat, targetLon, refLat, refLon, automaticNavigation
188         refLat = datahandler.latestLat
189         refLon = datahandler.latestLon
190         targetLat = lat
191         targetLon = lon
192         automaticNavigation = True
```

2 Rover Gumstix – Shell Scripts

2.1 start.sh

```
1  #!/bin/bash
2  export newfile=`ls /home/root/log/ | sed 's/_/ _/' | sort -rn | awk '{printf "%03d",
   $1 + 1; exit}'`
3  export DIR=/home/root/log/$newfile
4  mkdir $DIR
5  echo $DIR
6  /home/root/rover/camtest.sh &
7  /home/root/rover/mainprog.sh &
```

2.2 mainprog.sh

```
1  while [ 1 ]
2  do
3    python /home/root/rover/main.py > $DIR/main.log 2>&1 < /dev/null
4    sleep 30
5  done
```

2.3 wifi.sh

```
1  #!/bin/bash
2  dmesg -n1 #Quiten down driver
3  
4  #UART2
5  
6  devmem2 0x48002170 w 0x011C011C
7  devmem2 0x48002178 w 0x01180000
8  
9  #Web server
10 httpd -c /home/root/httpd.conf
11 
12 
13 #Weird initialisation sequence required for a slightly dodgy wireless driver
14 timeout 5 ifup wlan0
15 sleep 2
16 timeout 5 ifdown wlan0
17 sleep 2
18 timeout 60 ifup wlan0
19 /home/root/rover/start.sh > /home/root/start.sh.log
20 while [ 1 ]
```

```
do
      #Check if network is alive
       ping -w4 -c1 192.168.1.1
      rc=$?
      if [[ $rc != 0 ]] ; then
            echo Network failed
           timeout 5 ifdown wlan0
           sleep 3
            timeout 60 ifup wlan0
      fi
      sleep 10
done
```

3 Rover Arduino

3.1 rover.ino

```
#include "Wire.h"
#include "E2PROM.h"
#include "MPL3115A2.h"
#include "SerialPacket.h"
#include "board.h"
SerialPacket  sp;
MPL3115A2     tp;
E2PROM ee;
int lastMillis;
void errorBeep(const short data[]) {
      int length = data[0];
      for(int i = 1;i<length;i++) {
            tone(PIEZO,1500);
            if(data[i] == LONG_BEEP) {
                  delay(700);
            } else {
                  delay(300);
            };
            noTone(PIEZO);
            delay(250);
      };
};

//Take an average of multiple readings to improve stability and reduce errors.
float readADC_oversample(int pin) {
      long sum = 0;
      for(int i = 0;i < 5;i++) {
            sum += analogRead(pin);
      };
      return (sum / 5.0F);
};

//Return current battery voltage
float readBattery() {
      float vBat_raw = readADC_oversample(VBAT);
      float vBat = ((vBat_raw * VDD) / 1024.0F) * 2.0F * VBAT_CAL;
      return vBat;
};

```

```
40 //Run Power On Self Tests
41 void runTests() {
42       float vBat = readBattery();
43       Serial.println(F("# VBAT = "));
44        Serial.println(vBat,2);
45       if(vBat < VBAT_DEAD) {
46             while(1) {
47                   Serial.println(F("# VBAT is critical!!\n"));
48                   errorBeep(BEEP_BATTERY_DEAD);
49                   delay(1000);
50             };
51       };
52       if(vBat < VBAT_WARN) {
53             Serial.println(F("# VBAT is low!!\n"));
54             errorBeep(BEEP_BATTERY_LOW);
55             delay(500);
56       };
57
58       if(!tp.isPresent()) {
59             while(1) {
60                   Serial.println(F("# MPL3115A2 missing or dead!!\n"));
61                   errorBeep(BEEP_NO_TP);
62                   delay(1000);
63             };
64       };
65
66
67 };
68
69 void setup() {
70       Wire.begin();
71       sp.begin(9600);
72       tp.begin();
73        delay(100);
74        ee.begin_logging();
75       pinMode(STAT_LED,OUTPUT);
76       pinMode(LNK_RDY,OUTPUT);
77
78       pinMode(M1A_CTL,OUTPUT);
79       pinMode(M1B_CTL,OUTPUT);
80       pinMode(M2A_CTL,OUTPUT);
81       pinMode(M2B_CTL,OUTPUT);
82
83        pinMode(POD_MISO,INPUT_PULLUP);
84
85
86       digitalWrite(STAT_LED,HIGH);
87       //Power On Self Test
88       runTests();
89        lastMillis = millis();
90 };
91
92 void loop() {
93        int temp  = tp.readTemp()*10;
94        unsigned long pressure = tp.readPressure();
95       if(sp.isPacketAvailable()) {
96             char cmd[3];
```

```
97          sp.getCommand(cmd);
98          if(strncmp(cmd,"TP",2) == 0) { //Read temperature and pressure
99                char buffer[32];
100                   //Serial.println(temp);
101                   //Serial.println(pressure);
102                   snprintf(buffer,32,"%d %ld",temp,pressure);
103                   sp.sendReply("OK",buffer);
104             } else if(strncmp(cmd,"MT",2)==0) { //Motor control
105                   char params[64];
106                   sp.getPayload(params);
107                   bool commandOK = true;
108                   switch(params[0]) {
109                   case 'F':
110                         digitalWrite(M1A_CTL,HIGH);
111                         digitalWrite(M1B_CTL,LOW);
112                         break;
113                   case 'S':
114                         digitalWrite(M1A_CTL,LOW);
115                         digitalWrite(M1B_CTL,LOW);
116                         break;
117                   case 'B':
118                         digitalWrite(M1A_CTL,LOW);
119                         digitalWrite(M1B_CTL,HIGH);
120                         break;
121                   default:
122                         commandOK = false;
123                         break;
124                   }
125                   if(commandOK) {
126                         switch(params[2]) {
127                         case 'F':
128                               digitalWrite(M2A_CTL,HIGH);
129                               digitalWrite(M2B_CTL,LOW);
130                               break;
131                         case 'S':
132                               digitalWrite(M2A_CTL,LOW);
133                               digitalWrite(M2B_CTL,LOW);
134                               break;
135                         case 'B':
136                               digitalWrite(M2A_CTL,LOW);
137                               digitalWrite(M2B_CTL,HIGH);
138                               break;
139                         default:
140                               commandOK = false;
141                               break;
142                         }
143                   }
144                   if(commandOK) {
145                         sp.sendReply("OK","");
146                   } else {
147                         sp.sendReply("PE","");
148                   };
149             } else if(strncmp(cmd,"RD",2)==0) { //Dump EEPROM
150                   ee.begin_read();
151                   unsigned  char tmppage[7];
152                   while(!ee.is_end()) {
153                     ee.read_next_page(tmppage);
```

```
154                     if((tmppage[0] == 0x50) && (tmppage[1] == 0x41) &&
   (tmppage[2] == 0x47) && (tmppage[3] == 0x45) && (tmppage[4] == 0x00) &&
   (tmppage[5] == 0x00) &&  (tmppage[6] == 0x00)) {
155                       Serial.println("--BREAK--");
156                     } else {
157                       Serial.print((unsigned long)((unsigned long)tmppage[0] *
   65536L) + (unsigned long)(tmppage[1] * 256L) + tmppage[2]);
158                       Serial.print(",");
159                       Serial.print(((int)tmppage[3] << 8) | (int)tmppage[4]);
160                       Serial.print(",");
161                       Serial.println(((int)tmppage[5] << 8) | (int)tmppage[6]);
162                     };
163                   };
164               } else if(strncmp(cmd,"FT",2)==0) { //Format EEPROM (!!!)
165                     ee.format();
166                     sp.sendReply("OK","");
167               } else if(strncmp(cmd,"LK",2)==0) { //Poll deployment status
168                     if(digitalRead(POD_MISO)==LOW) {
169                       sp.sendReply("OK","N");
170                     } else {
171                       sp.sendReply("OK","D");
172                     };
173               } else if(strncmp(cmd,"PI",2)==0) {
174                     sp.sendReply("OK","");
175               } else { //Unknown command, return error status.
176                     sp.sendReply("CE","");
177               };
178         }
179         unsigned char eepage[7];
180
181         unsigned int deltat = millis() - lastMillis;
182         if(deltat > 2000) {
183           eepage[0] = (pressure >> 16L) & 0xFF;
184           eepage[1] = (pressure >> 8L) & 0xFF;
185           eepage[2] = (pressure) & 0xFF;
186           eepage[3] = (temp >> 8) & 0xFF;
187           eepage[4] = (temp) & 0xFF;
188           eepage[5] = (deltat >> 8) & 0xFF;
189           eepage[6] = (deltat) & 0xFF;
190           ee.write_next_page(eepage);
191           ee.update_header();
192           lastMillis = millis();
193         };
194         digitalWrite(STAT_LED,LOW); //Pulse LED as a 'heartbeat' indicator.
195         delay(150);
196         digitalWrite(STAT_LED,HIGH);
197         delay(150);
198   };
```

3.2 board.h

```
1   //Motors
2   #define M1A_CTL 6
3   #define M1B_CTL 9
4   #define M2A_CTL 10
5   #define M2B_CTL 5
6
7   //Analog inputs
```

```
8   #define VBAT A0
9
10  //Status monitoring
11  #define PIEZO 7
12  #define STAT_LED A2
13
14  //Communications
15  #define LNK_MOSI 0
16  #define LNK_MISO 1
17  #define LNK_RDY  A1
18
19  #define POD_MISO 4
20  #define POD_MOSI 8
21
22  //Supply voltage
23  #define VDD 4.8
24
25  //Battery measurement calibration
26  #define VBAT_CAL 1.0
27
28  #define VBAT_WARN 3.45
29  #define VBAT_DEAD 3.2
30
31  //Beeps
32  #define LONG_BEEP 1
33  #define SHORT_BEEP 0
34
35  const short int BEEP_BATTERY_LOW[] = {4,LONG_BEEP,SHORT_BEEP,SHORT_BEEP};
36  const short int BEEP_BATTERY_DEAD[] =
    {10,SHORT_BEEP,SHORT_BEEP,SHORT_BEEP,LONG_BEEP,LONG_BEEP,LONG_BEEP,SHORT_
    BEEP,SHORT_BEEP,SHORT_BEEP};
37
38  const short int BEEP_NO_TP[] =
    {10,SHORT_BEEP,SHORT_BEEP,LONG_BEEP,LONG_BEEP};
```

3.3 E2PROM.h

```
1   #include "Wire.h"
2   #include <Arduino.h>
3   /*EEPROM definitions
4   Devices split up into 'blocks' of 8 bytes
5   Device 0, block 0:
6   3 bytes start: 4e 4f 56
7   2 bytes page pointer
8   2 bytes unused (null for now)
9   1 byte checksum (see below)
10
11  'Page breaks' separate logging sessions
12  8 bytes: 50 41 47 45 00 00 00 2D
13
14  Data pages: 7 bytes data, 1 byte checksum
15  Data: 3 bytes pressure (Pa), 2 bytes temperature (C*10), 2 byte timestamp (millis since last log)
16  Checksum: 7*[Byte 0] + 6*[Byte 1]  + 5*[Byte 2] ... + 1*[Byte 7] modulo 256
17
18  Pressure & Vbat unsigned
19  Temperature signed
20
```

```
21 Page pointers:
22 Bits 0 is device ID
23 remaining bits are page on device
24 actual address = page id * 8
25 */
26 #define DEVICE_ADDRESS(b, n) (0x50 | b << 2 | n << 1)
27 class E2PROM {
28 private:
29       unsigned int current_write_page;
30       unsigned int current_read_page;
31       void write_byte(int device, unsigned long address, unsigned char data);
32       unsigned char read_byte(int device, unsigned long address);
33
34 public:
35       E2PROM(); //Constructor
36       void begin_logging(); //Initialise and insert 'page break'
37       void write_next_page(unsigned char *data); //Write 7 bytes of data + checksum
38       void write_page(unsigned int page, unsigned char *data); //Write an arbitrary
   page
39       bool read_page(unsigned int page, unsigned char *buffer); //Read an arbitrary
   page. True=ok, False=error
40       void begin_read(); //Reset read pointer
41       bool read_next_page(unsigned char *buffer); //Read next page (sequential reads).
   True=ok, False=error
42       bool is_end(); //True if end of EEPROM
43       unsigned int get_page_to_write(); //Get current page to write to
44       void update_header(); //Save write pointer to device 0 header
45       void format(); //Reset device 0 header
46 };
```

3.4 E2PROM.cpp

```
1  #include "E2PROM.h"
2  E2PROM::E2PROM() {
3       current_read_page = 0;
4  };
5
6  void E2PROM::begin_logging() {
7       unsigned char firstPage[7];
8       bool status = read_page(0, firstPage);
9       current_write_page = (firstPage[3] << 8) + firstPage[4];
10
11      //Check if devices are initialised
12      if((!status) || (firstPage[0] != 0x4e)) {
13            //Try again to be sure
14            status == read_page(0, firstPage);
15            //Must be blank, format
16            if((!status) || (firstPage[0] != 0x4e)) {
17                  unsigned char newPage[7] = {0x4e, 0x4f, 0x56,0x00, 0x01, 0x00,
   0x00};
18                  write_page(0, newPage);
19                   current_write_page = 1;
20            };
21      };
22      unsigned char page_break[7] = {0x50, 0x41, 0x47, 0x45, 0x00, 0x00, 0x00};
23      write_next_page(page_break);
24 };
25
```

```
26 unsigned char E2PROM::read_byte(int device, unsigned long address) {
27       Wire.beginTransmission((unsigned char)DEVICE_ADDRESS(address >>
   16,device));
28       Wire.write((address >> 8) & 0xFF); //High byte
29       Wire.write(address & 0xFF); //Low byte
30       Wire.requestFrom(DEVICE_ADDRESS(address >> 16,device),1);
31       Wire.endTransmission();
32
33       if(Wire.available()) {
34             return Wire.read();
35       } else {
36             return 0x00;
37       };
38 };
39
40 void E2PROM::write_byte(int device, unsigned long address, unsigned char data) {
41       Wire.beginTransmission((unsigned char)DEVICE_ADDRESS(address >>
   16,device));
42      //   Serial.print("WRITE AT: ");
43      //   Serial.println(address,HEX);
44       Wire.write((unsigned char)(address >> 8) & 0xFF); //High address byte
45       Wire.write((unsigned char)(address & 0xFF)); //Low address byte
46       Wire.write(data);
47       Wire.endTransmission();
48        delay(5);
49
50
51 };
52
53
54 bool E2PROM::read_page(unsigned int page, unsigned char *buffer) {
55       int byteCount = 0;
56       unsigned char readPage[8];
57       Wire.beginTransmission((unsigned char)DEVICE_ADDRESS((page >> 13) &
   0x01,page >> 14));
58       Wire.write((page >> 5) & 0xFF); //High address byte
59       Wire.write((page * 8) & 0xFF); //Low address byte
60       Wire.endTransmission();
61
62       Wire.requestFrom(DEVICE_ADDRESS((page >> 13) & 0x01,page >> 14),8);
63       while((Wire.available()) && (byteCount < 8)) {
64             readPage[byteCount] = Wire.read();
65         //     Serial.print("READ ox");
66 //Serial.println(readPage[byteCount],HEX);
67             if(byteCount < 7) buffer[byteCount] = readPage[byteCount];
68             byteCount++;
69       };
70
71       if(byteCount != 8) {
72             return false;
73       };
74       unsigned long tmpsum = 0;
75       for(int i = 0;i < 7;i++) {
76             tmpsum += (7 - i) * readPage[i];
77       };
78       unsigned char checksum;
79       checksum = tmpsum % 0xFF;
```

```
80      if(checksum != readPage[7]) {
81            return false;
82      } else {
83            return true;
84      };
85 };
86
87 void E2PROM::write_page(unsigned int page, unsigned char *data) {
88      unsigned long tmpsum = 0;
89      // Serial.print("WRITE AT PAGE: ");
90      // Serial.println(page,HEX);
91      for(int i = 0;i < 7;i++) {
92            tmpsum += (7 - i) * data[i];
93            write_byte(page >> 14, (page & 0x3FFF)*8  + i,data[i]);
94      };
95      unsigned char checksum;
96      checksum = tmpsum % 0xFF;
97      write_byte(page >> 14, (page & 0x3FFF)*8 + 7,checksum);
98 };
99
100    void E2PROM::write_next_page(unsigned char *data) {
101         if(current_write_page < 32768) {
102               write_page(current_write_page,data);
103               current_write_page++;
104         };
105    };
106
107    void E2PROM::begin_read() {
108         current_read_page = 1;
109    };
110
111    bool E2PROM::read_next_page(unsigned char *buffer) {
112         bool status = read_page(current_read_page,buffer);
113         current_read_page++;
114         return status;
115    };
116
117    bool E2PROM::is_end() {
118         if(current_read_page >= current_write_page) {
119               return true;
120         } else {
121               return false;
122         };
123    };
124
125    unsigned int E2PROM::get_page_to_write() {
126         return current_write_page;
127    };
128
129    void E2PROM::update_header() {
130         unsigned char newPage[7] = {0x4e, 0x4f, 0x56, 0x00, 0x00, 0x00, 0x00};
131         newPage[3] = current_write_page >> 8;
132         newPage[4] = current_write_page & 0xFF;
133         write_page(0, newPage);
134    };
135
136    void E2PROM::format() {
```

```
137         unsigned char newPage[7] = {0x4e, 0x4f, 0x56,0x00, 0x01, 0x00, 0x00};
138         write_page(0,newPage);
139         current_write_page = 1;
140         unsigned char page_break[7] = {0x50, 0x41, 0x47, 0x45, 0x00, 0x00,
    0x00};
141         write_next_page(page_break);
142     };
```

3.5 MPL3115A2.h (partly based on public domain code)

```
1   /*
2   Object-oriented MPL3115A2 Library. Based on open source code from Sparkfun.
3   */
4
5   #include <Wire.h> // for IIC communication
6   #include <Arduino.h>
7   //Constant definitions
8   #define STATUS 0x00
9   #define OUT_P_MSB 0x01
10  #define OUT_P_CSB 0x02
11  #define OUT_P_LSB 0x03
12  #define OUT_T_MSB 0x04
13  #define OUT_T_LSB 0x05
14  #define DR_STATUS 0x06
15  #define OUT_P_DELTA_MSB 0x07
16  #define OUT_P_DELTA_CSB 0x08
17  #define OUT_P_DELTA_LSB 0x09
18  #define OUT_T_DELTA_MSB 0x0A
19  #define OUT_T_DELTA_LSB 0x0B
20  #define WHO_AM_I 0x0C
21  #define F_STATUS 0x0D
22  #define F_DATA 0x0E
23  #define F_SETUP 0x0F
24  #define TIME_DLY 0x10
25  #define SYSMOD 0x11
26  #define INT_SOURCE 0x12
27  #define PT_DATA_CFG 0x13
28  #define BAR_IN_MSB 0x14
29  #define BAR_IN_LSB 0x15
30  #define P_TGT_MSB 0x16
31  #define P_TGT_LSB 0x17
32  #define T_TGT 0x18
33  #define P_WND_MSB 0x19
34  #define P_WND_LSB 0x1A
35  #define T_WND 0x1B
36  #define P_MIN_MSB 0x1C
37  #define P_MIN_CSB 0x1D
38  #define P_MIN_LSB 0x1E
39  #define T_MIN_MSB 0x1F
40  #define T_MIN_LSB 0x20
41  #define P_MAX_MSB 0x21
42  #define P_MAX_CSB 0x22
43  #define P_MAX_LSB 0x23
44  #define T_MAX_MSB 0x24
45  #define T_MAX_LSB 0x25
46  #define CTRL_REG1 0x26
47  #define CTRL_REG2 0x27
48  #define CTRL_REG3 0x28
```

```
#define CTRL_REG4 0x29
#define CTRL_REG5 0x2A
#define OFF_P 0x2B
#define OFF_T 0x2C
#define OFF_H 0x2D

#define MPL3115A2_ADDRESS 0x60 // 7-bit I2C address

class MPL3115A2 {
      public:
            void begin();
            bool isPresent();
            float readAltitude();
            float readPressure();
            float readTemp();

            void setModeBarometer();
            void setModeAltimeter();

            void setModeStandby();
            void setModeActive();
            void setFIFOMode(byte f_Mode);
            void setOversampleRate(byte sampleRate);
            void toggleOneShot(void);
            void enableEventFlags();
      private:
            byte IIC_Read(byte regAddr);
            void IIC_Write(byte regAddr, byte value);
};
```

3.6 MPL3115A2.cpp (partly based on public domain code)

```
#include "MPL3115A2.h"
void MPL3115A2::begin() {
  setModeBarometer(); // Measure pressure in Pascals from 20 to 110 kPa
  setOversampleRate(7); // Set Oversample to the recommended 128
  enableEventFlags(); // Enable all three pressure and temp event flags
};

//Returns the number of meters above sea level
float MPL3115A2::readAltitude()
{
  toggleOneShot(); //Toggle the OST bit causing the sensor to immediately take another reading

  //Wait for PDR bit, indicates we have new pressure data
  int counter = 0;
  while( (IIC_Read(STATUS) & (1<<1)) == 0)
  {
     if(++counter > 100) return(-999); //Error out
     delay(1);
  }

  // Read pressure registers
  Wire.beginTransmission(MPL3115A2_ADDRESS);
  Wire.write(OUT_P_MSB); // Address of data to get
  Wire.endTransmission(false); // Send data to I2C dev with option for a repeated start. THIS IS NECESSARY and not supported before Arduino V1.0.1!
  Wire.requestFrom(MPL3115A2_ADDRESS, 3); // Request three bytes
```

```
26
27   //Wait for data to become available
28   counter = 0;
29   while(Wire.available() < 3)
30   {
31     if(counter++ > 100) return(-999); //Error out
32     delay(1);
33   }
34
35   byte msb, csb, lsb;
36   msb = Wire.read();
37   csb = Wire.read();
38   lsb = Wire.read();
39
40   toggleOneShot(); //Toggle the OST bit causing the sensor to immediately take another reading
41
42   // The least significant bytes l_altitude and l_temp are 4-bit,
43   // fractional values, so you must cast the calulation in (float),
44   // shift the value over 4 spots to the right and divide by 16 (since
45   // there are 16 values in 4-bits).
46   float tempcsb = (lsb>>4)/16.0;
47
48   float altitude = (float)( (msb << 8) | csb) + tempcsb;
49
50   return(altitude);
51 }
52
53 //Reads the current pressure in Pa
54 //Unit must be set in barometric pressure mode
55 float MPL3115A2::readPressure()
56 {
57   toggleOneShot(); //Toggle the OST bit causing the sensor to immediately take another reading
58
59   //Wait for PDR bit, indicates we have new pressure data
60   int counter = 0;
61   while( (IIC_Read(STATUS) & (1<<2)) == 0)
62   {
63     if(++counter > 100) return(-999); //Error out
64     delay(1);
65   }
66
67   // Read pressure registers
68   Wire.beginTransmission(MPL3115A2_ADDRESS);
69   Wire.write(OUT_P_MSB); // Address of data to get
70   Wire.endTransmission(false); // Send data to I2C dev with option for a repeated start. THIS
   IS NECESSARY and not supported before Arduino V1.0.1!
71   Wire.requestFrom(MPL3115A2_ADDRESS, 3); // Request three bytes
72
73   //Wait for data to become available
74   counter = 0;
75   while(Wire.available() < 3)
76   {
77     if(counter++ > 100) return(-999); //Error out
78     delay(1);
79   }
80
81   byte msb, csb, lsb;
```

```
82   msb = Wire.read();
83   csb = Wire.read();
84   lsb = Wire.read();
85
86   toggleOneShot(); //Toggle the OST bit causing the sensor to immediately take another reading
87
88   // Pressure comes back as a left shifted 20 bit number
89   long pressure_whole = (long)msb<<16 | (long)csb<<8 | (long)lsb;
90   pressure_whole >>= 6; //Pressure is an 18 bit number with 2 bits of decimal. Get rid of
     decimal portion.
91
92   lsb &= 0b00110000; //Bits 5/4 represent the fractional component
93   lsb >>= 4; //Get it right aligned
94   float pressure_decimal = (float)lsb/4.0; //Turn it into fraction
95
96   float pressure = (float)pressure_whole + pressure_decimal;
97
98   return(pressure);
99 }
100
101    bool MPL3115A2::isPresent() {
102         if(IIC_Read(WHO_AM_I) == 196) return true;
103         else return false;
104    };
105
106    float MPL3115A2::readTemp()
107    {
108      toggleOneShot(); //Toggle the OST bit causing the sensor to immediately take another
       reading
109
110      //Wait for TDR bit, indicates we have new temp data
111      int counter = 0;
112      while( (IIC_Read(STATUS) & (1<<1)) == 0)
113      {
114        if(++counter > 100) return(-999); //Error out
115        delay(1);
116      }
117
118      // Read temperature registers
119      Wire.beginTransmission(MPL3115A2_ADDRESS);
120      Wire.write(OUT_T_MSB); // Address of data to get
121      Wire.endTransmission(false); // Send data to I2C dev with option for a repeated start.
       THIS IS NECESSARY and not supported before Arduino V1.0.1!
122      Wire.requestFrom(MPL3115A2_ADDRESS, 2); // Request two bytes
123
124      //Wait for data to become available
125      counter = 0;
126      while(Wire.available() < 2)
127      {
128        if(++counter > 100) return(-999); //Error out
129        delay(1);
130      }
131
132      byte msb, lsb;
133      msb = Wire.read();
134      lsb = Wire.read();
135
```

```
136     // The least significant bytes l_altitude and l_temp are 4-bit,
137     // fractional values, so you must cast the calulation in (float),
138     // shift the value over 4 spots to the right and divide by 16 (since
139     // there are 16 values in 4-bits).
140     float templsb = (lsb>>4)/16.0; //temp, fraction of a degree
141
142     float temperature = (float)(msb + templsb);
143
144     return(temperature);
145   }
146
147   //Sets the mode to Barometer
148   //CTRL_REG1, ALT bit
149   void MPL3115A2::setModeBarometer()
150   {
151     byte tempSetting = IIC_Read(CTRL_REG1); //Read current settings
152     tempSetting &= ~(1<<7); //Clear ALT bit
153     IIC_Write(CTRL_REG1, tempSetting);
154   }
155
156   //Sets the mode to Altimeter
157   //CTRL_REG1, ALT bit
158   void MPL3115A2::setModeAltimeter()
159   {
160     byte tempSetting = IIC_Read(CTRL_REG1); //Read current settings
161     tempSetting |= (1<<7); //Set ALT bit
162     IIC_Write(CTRL_REG1, tempSetting);
163   }
164
165   //Puts the sensor in standby mode
166   //This is needed so that we can modify the major control registers
167   void MPL3115A2::setModeStandby()
168   {
169     byte tempSetting = IIC_Read(CTRL_REG1); //Read current settings
170     tempSetting &= ~(1<<0); //Clear SBYB bit for Standby mode
171     IIC_Write(CTRL_REG1, tempSetting);
172   }
173
174   //Puts the sensor in active mode
175   //This is needed so that we can modify the major control registers
176   void MPL3115A2::setModeActive()
177   {
178     byte tempSetting = IIC_Read(CTRL_REG1); //Read current settings
179     tempSetting |= (1<<0); //Set SBYB bit for Active mode
180     IIC_Write(CTRL_REG1, tempSetting);
181   }
182
183   //Setup FIFO mode to one of three modes. See page 26, table 31
184   //From user jr4284
185   void MPL3115A2::setFIFOMode(byte f_Mode)
186   {
187     if (f_Mode > 3) f_Mode = 3; // FIFO value cannot exceed 3.
188     f_Mode <<= 6; // Shift FIFO byte left 6 to put it in bits 6, 7.
189
190     byte tempSetting = IIC_Read(F_SETUP); //Read current settings
191     tempSetting &= ~(3<<6); // clear bits 6, 7
```

```
192     tempSetting |= f_Mode; //Mask in new FIFO bits
193     IIC_Write(F_SETUP, tempSetting);
194   }
195
196   //Call with a rate from 0 to 7. See page 33 for table of ratios.
197   //Sets the over sample rate. Datasheet calls for 128 but you can set it
198   //from 1 to 128 samples. The higher the oversample rate the greater
199   //the time between data samples.
200   void MPL3115A2::setOversampleRate(byte sampleRate)
201   {
202     if(sampleRate > 7) sampleRate = 7; //OS cannot be larger than 0b.0111
203     sampleRate <<= 3; //Align it for the CTRL_REG1 register
204
205     byte tempSetting = IIC_Read(CTRL_REG1); //Read current settings
206     tempSetting &= 0b11000111; //Clear out old OS bits
207     tempSetting |= sampleRate; //Mask in new OS bits
208     IIC_Write(CTRL_REG1, tempSetting);
209   }
210
211   //Clears then sets the OST bit which causes the sensor to immediately take another reading
212   //Needed to sample faster than 1Hz
213   void MPL3115A2::toggleOneShot(void)
214   {
215     byte tempSetting = IIC_Read(CTRL_REG1); //Read current settings
216     tempSetting &= ~(1<<1); //Clear OST bit
217     IIC_Write(CTRL_REG1, tempSetting);
218
219     tempSetting = IIC_Read(CTRL_REG1); //Read current settings to be safe
220     tempSetting |= (1<<1); //Set OST bit
221     IIC_Write(CTRL_REG1, tempSetting);
222   }
223
224   //Enables the pressure and temp measurement event flags so that we can
225   //test against them. This is recommended in datasheet during setup.
226   void MPL3115A2::enableEventFlags()
227   {
228     IIC_Write(PT_DATA_CFG, 0x07); // Enable all three pressure and temp event flags
229   }
230
231   // These are the two I2C functions in this sketch.
232   byte MPL3115A2::IIC_Read(byte regAddr)
233   {
234     // This function reads one byte over IIC
235     Wire.beginTransmission(MPL3115A2_ADDRESS);
236     Wire.write(regAddr); // Address of CTRL_REG1
237     Wire.endTransmission(false); // Send data to I2C dev with option for a repeated start.
    THIS IS NECESSARY and not supported before Arduino V1.0.1!
238     Wire.requestFrom(MPL3115A2_ADDRESS, 1); // Request the data...
239     return Wire.read();
240   }
241
242   void MPL3115A2::IIC_Write(byte regAddr, byte value)
243   {
244     // This function writes one byto over IIC
245     Wire.beginTransmission(MPL3115A2_ADDRESS);
246     Wire.write(regAddr);
```

```
247       Wire.write(value);
248       Wire.endTransmission(true);
249     }
```

3.7 SerialPacket.h

```
1   #ifndef SerialPacket_h
2   #define SerialPacket_h
3   #include "stdlib.h"
4   #include "string.h"
5   #include "Arduino.h"
6   class SerialPacket
7   {
8   public:
9         SerialPacket();
10        bool isPacketAvailable();
11        void getCommand(char *buffer);
12        void getPayload(char *buffer);
13        void sendReply(char *status,char *payload);
14        void begin(int baud);
15  private:
16        char currentPacket[64];
17        bool packetAvailable;
18  };
```

3.8 SerialPacket.cpp

```
1   #include "Arduino.h"
2   #include "SerialPacket.h"
3
4   SerialPacket::SerialPacket() {
5         packetAvailable = false;
6   };
7
8   /*
9   Initialises the serial port.
10  */
11  void SerialPacket::begin(int baud) {
12        Serial.begin(baud);
13  }
14
15  /*
16  Run regularly to check for incoming packets.
17  Returns false if no packet available.
18  Returns true if a packet was received.
19  */
20  bool SerialPacket::isPacketAvailable() {
21        char incomingPacket[64] = {0};
22        int currentLocation = 0;
23        int startCounter = 0;
24        char startBytes[] = {'H','I',' '}; //Set of characters signalling start of transmission
25        char cByte;
26        while(Serial.available() > 0) {
27              cByte = Serial.read();
28              //Serial.print("Read byte: ");
29              //Serial.println(cByte);
30              if (startCounter < 3) {
31                    if (cByte == startBytes[startCounter]) {
```

```
32                      startCounter++;
33                  } else {
34                      startCounter = 0;
35                  };
36              } else {
37                  if (cByte == '\n') { //Check for terminating character
38                      packetAvailable = true;
39                      strncpy(currentPacket,incomingPacket,64);
40                      return true; //If so, finish up and return that a packet was received
41                  } else  {
42                      incomingPacket[currentLocation] = cByte;
43                      currentLocation++;
44                  }
45              };
46              delayMicroseconds(2500); //Wait for the next byte, just to prevent errors occuring
    in some situations.
47      };
48      return false;
49  };
50
51  /*
52  Fetches the command part of the received packet.
53  Buffer must be char array length > 2
54  */
55  void SerialPacket::getCommand(char *buffer) {
56      strncpy(buffer,currentPacket,2);//First two bytes are command
57  };
58
59  /*
60  Fetches the payload part of the received packet.
61  Buffer must be char array length > 64
62  */
63  void SerialPacket::getPayload(char *buffer) {
64      strncpy(buffer,&(currentPacket[3]),64);//Everything from third byte onwards is the
    payload
65  };
66
67  /*
68  Sends a reply to the master.
69  status: string containing 2-letter status code
70  payload: string containing the payload
71  */
72  void SerialPacket::sendReply(char *status,char *payload) {
73      Serial.print("RP ");
74      Serial.print(status);
75      Serial.print(" ");
76      Serial.print(payload);
77      Serial.println("\n");
78  };
```

4 Landing Module Arduino

4.1 landingModule.ino

```
1   #include <I2Cdev.h>
2   #include <strings.h>
3   #include <helper_3dmath.h>
```

```
4  #include <MPU6050.h>
5
6  #include <ArduinoStream.h>
7  #include <bufstream.h>
8  #include <ios.h>
9  #include <iostream.h>
10 #include <istream.h>
11 #include <MinimumSerial.h>
12 #include <ostream.h>
13 #include <Sd2Card.h>
14 #include <SdBaseFile.h>
15 #include <SdFat.h>
16 #include <SdFatConfig.h>
17 #include <SdFatmainpage.h>
18 #include <SdFatStructs.h>
19 #include <SdFatUtil.h>
20 #include <SdFile.h>
21 #include <SdInfo.h>
22 #include <SdSpi.h>
23 #include <SdStream.h>
24 #include <SdVolume.h>
25
26 #include <Wire.h>
27
28 #define ADDRESS 0x77
29
30 uint32_t D1 = 0;
31 uint32_t D2 = 0;
32
33 int32_t dT = 0;
34 int32_t TEMP = 0;
35 int64_t OFF = 0;
36 int64_t SENS = 0;
37 int32_t P = 0;
38
39 uint16_t C[7];
40 const int chipSelect = 4;
41 SdFat sd;
42 SdFile myFile;
43 char filename[14];
44
45 MPU6050 accelgyro;
46
47 int16_t ax, ay, az;
48 int16_t gx, gy, gz;
49 bool gpsAvailable = false;
50 double latitude = 0, longitude = 0, gpsAlt = 0;
51 int gpshr = 0, gpsmin = 0, gpssec = 0;
52
53 enum deployment_state {
54   WAITING = 0,
55   LAUNCHING = 1,
56   LAUNCHED = 2,
57   LANDING = 3,
58   LANDED = 4,
59   READY_TO_DEPLOY = 5,
60   DEPLOYING = 6,
```

```
61   DEPLOYED = 7,
62   CANCELLED = 8
63 };
64
65 deployment_state currentState = WAITING;
66
67 #define AUX 20
68 #define SET 27
69 #define PULLUP 26
70 #define M2A_CTL 14
71 #define M2B_CTL 15
72
73 #define MS_OPEN 26
74
75 bool useSD = false;
76
77 void openLatch() {
78   long startTime = millis();
79   digitalWrite(M2A_CTL,HIGH);
80   digitalWrite(M2B_CTL,LOW);
81   while((digitalRead(MS_OPEN) == 1) && ((millis() - startTime) < 7500));
82   digitalWrite(M2A_CTL,LOW);
83   digitalWrite(M2B_CTL,LOW);
84 };
85
86
87 void setup() {
88     currentState = WAITING;
89     pinMode(PULLUP,OUTPUT);
90     digitalWrite(PULLUP,HIGH);
91     pinMode(MS_OPEN,INPUT); //NB: 1 = switch inactive, 0 = switch active
92     pinMode(M2A_CTL,OUTPUT);
93     pinMode(M2B_CTL,OUTPUT);
94     pinMode(28,OUTPUT);
95     pinMode(AUX, OUTPUT);
96     pinMode(SET, OUTPUT);
97     Wire.begin();
98     Serial.begin(9600);
99     Serial1.begin(9600);
100         digitalWrite(SET,HIGH);
101         delay(100);
102         digitalWrite(SET,LOW);
103         /*
104        433.2 MHz (433200)
105        RF 19200 baud (4)
106        20mW output power (9)
107        UART 9600 baud (3)
108        No parity (0)
109        */
110         delay(2);
111         Serial.println("WR_433200_4_9_3_0");
112         delay(200);
113         digitalWrite(SET,HIGH);
114         delay(200);
115         Serial.println("#Initialising I2C devices");
116         initial(ADDRESS);
117         accelgyro.initialize();
```

```
118       Serial.println("#Initialised I2C devices");
119
120       Serial.println("#Initialising uSD");
121       if (!sd.begin(chipSelect, SPI_SIXTEENTH_SPEED)) {
122         //sd.initErrorHalt();
123            useSD = false;
124           Serial.println("---SD ERROR---");
125       } else {
126         Serial.println("#Initialised uSD");
127         for(int i = 0;i<999;i++) {
128           snprintf(filename,14,"log%03d.csv",i);
129           if(!sd.exists(filename)) break;
130         };
131         Serial.print("#Writing to ");
132         Serial.println(filename);
133         if (!myFile.open(filename, O_RDWR | O_CREAT | O_AT_END)) {
134           //sd.errorHalt("#Opening file for write failed");
135         }
136         myFile.println("time,t,p,gps,lat,lon,alt,ax,ay,az,gx,gy,gz");
137         myFile.close();
138       }
139   }
140   long t = 0;
141   char serialBuffer[500] = {0};
142   char *bufferPosition = serialBuffer;
143   long nullPressure = -1;
144
145   float altitudeStateStore = 0;
146   long timeStateStore = 0;
147   char ch1 = 0, ch2 = 0, ch3 = 0;
148   void loop() {
149      if((millis() - t) > 600) {
150         digitalWrite(28,HIGH);
151         D1 = getVal(ADDRESS, 0x48); // Pressure raw
152         D2 = getVal(ADDRESS, 0x58);// Temperature raw
153
154         dT = (D2 - C[4] * 256);
155         TEMP = 2000 + dT * C[5]/8388608;
156         OFF  = C[1] * 65536LL + (dT *C[3])/128;
157         SENS = C[0] * 32768LL + (C[2] * dT)/256;
158         P = (D1 * SENS/2097152 - OFF)/32768;
159         if(nullPressure == -1) nullPressure = P;
160         t = millis();
161
162         accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);
163
164         digitalWrite(AUX,HIGH);
165         delay(20);
166         Serial.print("START");
167         Serial.print(TEMP);
168         Serial.print(",");
169         Serial.print(P);
170         Serial.print(",");
171         Serial.print(gpsAvailable);
172         Serial.print(",");
173         Serial.print(latitude,8);
174         Serial.print(",");
```

```
175         Serial.print(longitude,8);
176         Serial.print(",");
177         Serial.print(gpsAlt);
178         Serial.print(",");
179         Serial.print(currentState);
180         Serial.print(",");
181         Serial.print(millis() - timeStateStore);
182         Serial.print(",");
183
184         Serial.print(relativisePressure(P));
185         Serial.println("END");
186         delay(10);
187         digitalWrite(AUX,LOW);
188        // digitalWrite(AUX,LOW);
189         if (!myFile.open(filename, O_RDWR | O_CREAT | O_AT_END)) {
190           //sd.errorHalt("#Opening file for write failed");
191           useSD = false;
192           Serial.println("---SD ERROR---");
193         } else {
194           myFile.print(t);
195           myFile.print(",");
196           myFile.print(TEMP);
197           myFile.print(",");
198           myFile.print(P);
199           myFile.print(",");
200           myFile.print(gpsAvailable);
201           myFile.print(",");
202           myFile.print(latitude,8);
203           myFile.print(",");
204           myFile.print(longitude,8);
205           myFile.print(",");
206           myFile.print(gpsAlt);
207           myFile.print(",");
208           myFile.print(ax);
209           myFile.print(",");
210           myFile.print(ay);
211           myFile.print(",");
212           myFile.print(az);
213           myFile.print(",");
214           myFile.print(gx);
215           myFile.print(",");
216           myFile.print(gy);
217           myFile.print(",");
218           myFile.println(gz);
219           myFile.close();
220         };
221        // delay(100);
222         digitalWrite(28,LOW);
223       //  delay(400);
224       }
225       while(Serial1.available()) {
226         *bufferPosition = Serial1.read();
227         if(*bufferPosition == '\n') {
228           *(++bufferPosition) = '\0';
229           Serial.println(serialBuffer);
230           //Serial.print("END");
231           parseNMEA();
```

```
232           bufferPosition = serialBuffer;
233         } else {
234           bufferPosition++;
235           if(bufferPosition >= (serialBuffer+500)) {
236             bufferPosition = serialBuffer;
237           }
238         };
239       };
240       switch(currentState) {
241        case WAITING:
242          if(relativisePressure(P) > 50) {
243            currentState = LAUNCHING;
244            timeStateStore = millis();
245            altitudeStateStore = relativisePressure(P);
246          };
247          break;
248        case LAUNCHING:
249          if((millis() - timeStateStore) > 10000) {
250            if(abs(altitudeStateStore - relativisePressure(P)) < 5) {
251              currentState = LAUNCHED;
252            };
253            if((altitudeStateStore - relativisePressure(P)) > 25) {
254              currentState = LANDING;
255            };
256            altitudeStateStore = relativisePressure(P);
257            timeStateStore = millis();
258          };
259          break;
260        case LAUNCHED:
261          if((altitudeStateStore - relativisePressure(P)) > 50) {
262             currentState = LANDING;
263             altitudeStateStore = relativisePressure(P);
264             timeStateStore = millis();
265          };
266          break;
267         case LANDING:
268          if((millis() - timeStateStore) > 30000) {
269            if(abs(altitudeStateStore - relativisePressure(P)) < 5) {
270              currentState = LANDED;
271            };
272            altitudeStateStore = relativisePressure(P);
273            timeStateStore = millis();
274          };
275          break;
276         case LANDED:
277          if((millis() - timeStateStore) > 30000) {
278            currentState = READY_TO_DEPLOY;
279          };
280          break;
281         case READY_TO_DEPLOY:
282          if((millis() - timeStateStore) > 90000) {
283            currentState = DEPLOYING;
284          };
285          break;
286         case DEPLOYING:
287          //Motor and microswitch stuff
288          openLatch();
```

```
289           currentState = DEPLOYED;
290           break;
291         case DEPLOYED:
292           //Nothing here for now
293           break;
294       }
295       while(Serial.available() > 0) {
296         ch1 = ch2;
297         ch2 = ch3;
298         ch3 = Serial.read();
299         if((ch1 == 'E') && (ch2 == 'N') && (ch3 == 'D')) {
300           currentState = CANCELLED;
301         };
302         if((ch1 == 'D') && (ch2 == 'E') && (ch3 == 'P')) {
303           currentState = DEPLOYING;
304         };
305       };
306     };
307
308     inline int chrToInt(char c) {
309      return c - '0';
310     }
311     //Convert pressure to relative altitude
312     float relativisePressure(float Pr) {
313       float altitude;
314       if(nullPressure == -1) return 0;
315       altitude = 44330.0 * (1.0 - pow(Pr / nullPressure,0.1903));
316       return altitude;
317     }
318
319     void parseNMEA() {
320       char splitstr[18][15];
321       int currentPoint = 0;
322       //Only parse position updates
323       if(strncmp(serialBuffer,"$GPGGA",6)==0) {
324         //Disassemble NMEA sentence
325         int index = 0, index2 = 0;
326         char currentChar  = serialBuffer[0];
327         while((currentChar != '\0') && (currentPoint < 18) && (index < 500)) {
328           if((currentChar == ',') || (index2 > 14)) {
329             splitstr[currentPoint][index2] = '\0';
330             currentPoint++;
331             index2 = 0;
332           } else {
333             splitstr[currentPoint][index2] = currentChar;
334             index2++;
335           };
336           index++;
337           currentChar = serialBuffer[index];
338         };
339         if(currentPoint < 13) {
340           Serial.println("Rejecting bad GPGGA");
341           return;
342         };
343         gpshr = chrToInt(splitstr[1][0]) * 10 + chrToInt(splitstr[1][1]);
344         gpsmin = chrToInt(splitstr[1][2]) * 10 + chrToInt(splitstr[1][3]);
345         gpssec = chrToInt(splitstr[1][4]) * 10 + chrToInt(splitstr[1][5]);
```

```
346       int latDegrees = chrToInt(splitstr[2][0]) * 10 + chrToInt(splitstr[2][1]);
347       float latMinutes = chrToInt(splitstr[2][2]) * 10.0 + chrToInt(splitstr[2][3]) *
    1.0 + chrToInt(splitstr[2][5]) * 0.1 +  chrToInt(splitstr[2][6]) * 0.01 +
    chrToInt(splitstr[2][7]) * 0.001 +  chrToInt(splitstr[2][8]) * 0.0001;
348       latitude = latDegrees + (latMinutes / 60.0);
349
350       if(splitstr[3][0]=='S') {
351         latitude = -latitude;
352       };
353
354       int lonDegrees = chrToInt(splitstr[4][0]) * 100 + chrToInt(splitstr[4][1]) * 10
    + chrToInt(splitstr[4][2]);
355       float lonMinutes = chrToInt(splitstr[4][3]) * 10.0 + chrToInt(splitstr[4][4]) *
    1.0 + chrToInt(splitstr[4][6]) * 0.1 +  chrToInt(splitstr[4][7]) * 0.01 +
    chrToInt(splitstr[4][8]) * 0.001 +  chrToInt(splitstr[4][9]) * 0.0001;
356       longitude = lonDegrees + (lonMinutes / 60.0);
357       if(splitstr[5][0]=='W') {
358         longitude = -longitude;
359       };
360
361       gpsAlt = atof(splitstr[9]);
362       Serial.println(splitstr[7]);
363       if(chrToInt(splitstr[6][0])>0) {
364        gpsAvailable = true;
365       } else {
366        gpsAvailable = false;
367       }
368       };
369    };
370
371    long getVal(int address, byte code)
372    {
373       unsigned long ret = 0;
374       Wire.beginTransmission(address);
375       Wire.write(code);
376       Wire.endTransmission();
377       delay(10);
378       // start read sequence
379       Wire.beginTransmission(address);
380       Wire.write((byte) 0x00);
381       Wire.endTransmission();
382       Wire.beginTransmission(address);
383       Wire.requestFrom(address, (int)3);
384       if(Wire.available()) {
385       //    ret = ((uint32_t)Wire.read() << 16) | ((uint32_t)Wire.read() << 8) | Wire.read();
386             ret = Wire.read() * 65536 + Wire.read() * 256 + Wire.read();
387       }
388       else {
389          ret = -1;
390       }
391       Wire.endTransmission();
392       return ret;
393    }
394
395    void initial(uint8_t address)
396    {
397       Serial.println();
```

```
398         Serial.println("#PROM COEFFICIENTS");
399
400         Wire.beginTransmission(address);
401         Wire.write(0x1E); // reset
402         Wire.endTransmission();
403         delay(10);
404
405         for (int i=0; i<7; i++) {
406             Wire.beginTransmission(address);
407             Wire.write(0xA2 + (i * 2));
408             Wire.endTransmission();
409
410             Wire.beginTransmission(address);
411             Wire.requestFrom(address, (uint8_t) 6);
412             delay(1);
413             if(Wire.available()) {
414                 C[i] = Wire.read() *256 + Wire.read();
415             }
416             else {
417                 Serial.println("#Error reading PROM 1"); // error reading the PROM or
    communicating with the device
418             }
419             Serial.println(C[i]);
420         }
421         Serial.println();
422     }
```

5 Base Station Arduino

5.1 baseStation.ino

```
1   #include <LSM303.h>
2   #include <Adafruit_ST7735.h>
3   #include <Adafruit_GFX.h>
4   #include <SPI.h>
5   #include <Wire.h>
6   #include <strings.h>
7   #include "lcd.h"
8
9   /*--PIN DEFINITIONS--*/
10  //LCD
11  #define cs   53
12  #define dc   46
13  #define rst  44
14  //RF
15  #define SET 16
16  #define AUX 17
17  /*--------------------*/
18
19  int htol = 2, vtol = 5; //Antenna position tolerance
20
21  LSM303 compass;
22  Adafruit_ST7735 tft = Adafruit_ST7735(cs, dc, rst);
23
24  float overSampleHeading() {
25      float heading = 0;
26    for(int i = 0;i<10;i++) {
```

```
27    compass.read();
28    heading += compass.heading();
29    delay(1);
30   };
31  // Serial.println(heading / 10);
32   return (heading / 10.0);
33 };
34
35 //Display 'Team Nova' logo with text below
36 void drawSplash(char *text) {
37   tft.fillScreen(ST7735_WHITE);
38
39  /// testlines(tft.Color565(255, 0, 255));
40   int address = 0;
41   for(int i = 0; i < 41; i++) { //41 lines
42     for(int j = 0;j<120;j++) { //120 cols
43       tft.drawPixel(j+4, i+10,tft.Color565(pgm_read_byte_near(pixel_data+address),
44                                            pgm_read_byte_near(pixel_data+address+1),
45                                            pgm_read_byte_near(pixel_data+address+2)));
46     // tft.drawPixel(j+4,i+10,tft.Color565(255,0,0));
47       address += 3;
48       //delay(1);
49      };
50   };
51   tft.setTextColor(ST7735_BLACK);
52   //FIXME - center text on screen
53   tft.setCursor(23,80);
54   tft.setTextSize(2);
55   tft.println(text);
56 };
57
58 //For drawing position guidance arrows
59 void drawArrows(boolean up, boolean down, boolean left, boolean right, uint16_t color) {
60   if(up) {
61     tft.fillTriangle(48,65,80,65,64,33,color);
62   };
63   if(down) {
64     tft.fillTriangle(48,95,80,95,64,127,color);
65   };
66   if(left) {
67     tft.fillTriangle(48,65,48,95,16,80,color);
68   };
69    if(right) {
70     tft.fillTriangle(80,65,80,95,112,80,color);
71   };
72 };
73
74 void setup() {
75   pinMode(19,INPUT);
76   Serial1.begin(9600);
77   Serial.begin(57600);
78   pinMode(AUX, OUTPUT);
79   pinMode(SET, OUTPUT);
80   digitalWrite(AUX,LOW);
81
82
83   digitalWrite(SET,HIGH);
```

```
84   Wire.begin();
85   compass.init();
86   compass.enableDefault();
87   //If we have time, change these for maximum accuracy. Probably not critical though.
88   compass.m_min = (LSM303::vector<int16_t>){-32767, -32767, -32767};
89   compass.m_max = (LSM303::vector<int16_t>){+32767, +32767, +32767};
90
91   tft.initR(INITR_BLACKTAB);
92   tft.fillScreen(ST7735_WHITE);
93   delay(100);
94   digitalWrite(SET,LOW);
95   delay(1);
96   /*
97  433.2 MHz (433200)
98  RF 19200 baud (4)
99  20mW output power (9)
100      UART 9600 baud (3)
101      No parity (0)
102      */
103      Serial1.print("WR_433200_4_9_3_0");
104      Serial1.write(0x0D);
105      Serial1.write(0X0A);
106      delay(200);
107      digitalWrite(SET,HIGH);
108      delay(200);
109      drawSplash("OFFLINE");
110    };
111
112    char radioBuffer[1500];
113    char *endOfBuffer = radioBuffer;
114    char commandBuffer[80];
115    char *endOfCommandBuffer = commandBuffer;
116    long timeLastUpdate = 0;
117
118    int angleV, angleH;
119
120    /*
121    *0: offline
122    *1: no data
123    *2: tracking
124    */
125    int state = 0, lastState = 0;
126
127    void loop() {
128
129      //Check for available data on PC interface
130
131      while(Serial.available() > 0) {
132        if(state == 0) {
133          state = 1;
134        };
135        *endOfCommandBuffer = Serial.read();
136        //Serial.write(*endOfCommandBuffer);
137
138        if(*(endOfCommandBuffer) == '#') {
139          //endOfCommandBuffer++;
```

```
140         *endOfCommandBuffer = '\0';
141         //Handle serial command
142         if(strncmp(commandBuffer,"A",1) == 0) { //Set antenna position
143           char *part1 = strtok(commandBuffer," ");
144           part1 = strtok(NULL," ");
145           //Serial.println(part1);
146           char *part2 = strtok(NULL," ");
147          // Serial.println(part2);
148           angleH = atoi(part1);
149           angleV = atoi(part2);
150           state = 2;
151         }
152         else if(strncmp(commandBuffer,"R",1) == 0) { //Read out radio buffer
153           for(char *c = radioBuffer;c<endOfBuffer;c++) {
154             Serial.write(*c);
155           };
156           Serial.println();
157           Serial.println("-----END-----");
158           endOfBuffer = radioBuffer;
159         }
160         else if(strncmp(commandBuffer,"W",1) == 0) { //Transmit data over radio
161           digitalWrite(AUX,HIGH);
162           delay(2);
163           Serial1.print(commandBuffer+2);
164           delay(2);
165           digitalWrite(AUX,LOW);
166         };
167         endOfCommandBuffer = commandBuffer;
168       } else {
169         endOfCommandBuffer++;
170       }
171       //Wrap around in event of buffer overflow
172       if(endOfCommandBuffer == commandBuffer+80) {
173         endOfCommandBuffer = commandBuffer;
174       };
175     };
176       //Check for available data on RF interface
177     while(Serial1.available() > 0) {
178
179       *endOfBuffer = Serial1.read();
180       endOfBuffer++;
181       //Wrap around in event of buffer overflow
182       if(endOfBuffer >= radioBuffer+1500) {
183         endOfBuffer = radioBuffer;
184       };
185     };
186     if((millis() - timeLastUpdate) > 750) {
187       //Update display if appropriate
188       timeLastUpdate = millis();
189
190       if((state == 1) && (lastState == 0)) {
191         drawSplash("NO DATA");
192       };
193       if(state == 2) {
194         compass.read();
195         int currentH = (int)overSampleHeading();
196         /*Serial.print("X=");
```

```
197         Serial.print(compass.a.x);
198         Serial.print(" Y=");
199         Serial.print(compass.a.y);
200         Serial.print(" Z=");
201         Serial.println(compass.a.z);*/
202
203           int currentV = atan(compass.a.y/9800.0) * (180.0 / 3.14152); //TODO:
    accelerometer read
204           bool up = false, down = false, left = false, right = false;
205           tft.fillScreen(ST7735_WHITE);
206           int deltaH = ((currentH - angleH + 360) % 360);
207           int deltaV = ((currentV - angleV + 360) % 360);
208           if((deltaH < 180) && (deltaH > htol)) {
209             right = true;
210           };
211           if((deltaH > 180) && (deltaH < 360-htol)) {
212             left = true;
213           };
214
215           if((deltaV < 180) && (deltaV > vtol)) {
216             up = true;
217           };
218           if((deltaV > 180) && (deltaV < 360-vtol)) {
219             down = true;
220           };
221           drawArrows(up,down,left,right,ST7735_RED);
222         };
223             lastState = state;
224       };
225     };
```

6 Base Station PC – Java

6.1 Main.java

```
1  package gui;
2
3
4  import gui.NetworkRover.MoveOperation;
5
6  import java.awt.EventQueue;
7
8  import javax.swing.JFrame;
9  import javax.swing.JLabel;
10 import javax.swing.JOptionPane;
11 import javax.swing.JScrollPane;
12 import javax.swing.SpringLayout;
13 import javax.swing.JPanel;
14 import javax.swing.UIManager;
15
16 import java.awt.Color;
17 import java.awt.Component;
18
19 import javax.swing.Box;
20
21 import java.awt.FlowLayout;
22 import java.awt.event.ActionEvent;
```

```
23 import java.awt.event.ActionListener;
24 import java.nio.file.Files;
25 import java.nio.file.Paths;
26 import java.util.concurrent.Executors;
27 import java.util.concurrent.ScheduledExecutorService;
28 import java.util.concurrent.TimeUnit;
29
30 import javax.swing.JButton;
31 import javax.swing.JCheckBox;
32
33
34 public class Main {
35
36       private JFrame frame;
37    private CustomMap map_2;
38    private SettingsStore settings;
39    private NetworkRover rover;
40    private ScheduledExecutorService updater,  updater1, updater2;
41    @SuppressWarnings("unused")
42       private LargeMap m;
43    private LandingModule module;
44    private BaseStationCommunications comms;
45    private AntennaManager ant;
46    private DataLogger d;
47       /**
48     * Launch the application.
49     */
50
51       public static void main(String[] args) {
52             EventQueue.invokeLater(new Runnable() {
53                   public void run() {
54                         try {
55                               Main window = new Main();
56                               window.frame.setVisible(true);
57
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
58                         } catch (Exception e) {
59                               e.printStackTrace();
60                         }
61                   }
62             });
63       }
64
65       /**
66      * Create the application.
67      */
68
69       public Main() {
70             initialize();
71       }
72
73       /**
74      * Initialise the contents of the frame.
75      */
76       private void initialize() {
77             settings = new SettingsStore();
78
```

```
79              module = new LandingModule();
80              comms = new BaseStationCommunications();
81              ant = new AntennaManager();
82              frame = new JFrame();
83              frame.setBounds(0,0,1280,739);
84              frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
85              frame.setTitle("Nova | Mission Control");
86              SpringLayout springLayout = new SpringLayout();
87              frame.getContentPane().setLayout(springLayout);
88
89              /*JPanel panel = new JPanel();
90          panel.setBackground(Color.BLACK);
91          panel.setBorder(new SoftBevelBorder(BevelBorder.LOWERED, null, null, null, null));
92          springLayout.putConstraint(SpringLayout.NORTH, panel, 10, SpringLayout.NORTH,
    frame.getContentPane());
93          springLayout.putConstraint(SpringLayout.WEST, panel, 10, SpringLayout.WEST,
    frame.getContentPane());
94          springLayout.putConstraint(SpringLayout.SOUTH, panel, -10, SpringLayout.SOUTH,
    frame.getContentPane());
95          springLayout.putConstraint(SpringLayout.EAST, panel, 300, SpringLayout.WEST,
    frame.getContentPane());
96          frame.getContentPane().add(panel);*/
97
98              /*JTextArea txtrDebugArea = new JTextArea();
99          txtrDebugArea.setFont(new Font("Lucida Console", Font.PLAIN, 12));
100             txtrDebugArea.setForeground(Color.GREEN);
101             txtrDebugArea.setBackground(Color.BLACK);
102             txtrDebugArea.setEditable(false);
103             txtrDebugArea.setText("Debug Area\nPlaceholder");*/
104                 //final Console console = new Console();
105
106                 //Current console data should be in raw, not debug - change null in line below
    back to as appropriate later.
107                 JScrollPane panel = new JScrollPane(null,
108                             JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
109                             JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
110                 springLayout.putConstraint(SpringLayout.NORTH, panel, 10,
    SpringLayout.NORTH, frame.getContentPane());
111                 springLayout.putConstraint(SpringLayout.WEST, panel, 10,
    SpringLayout.WEST, frame.getContentPane());
112                 springLayout.putConstraint(SpringLayout.SOUTH, panel, -10,
    SpringLayout.SOUTH, frame.getContentPane());
113                 springLayout.putConstraint(SpringLayout.EAST, panel, 300,
    SpringLayout.WEST, frame.getContentPane());
114                 frame.getContentPane().add(panel);
115                 //panel.add(consoleContainer);
116
117                 Component horizontalStrut = Box.createHorizontalStrut(10);
118                 springLayout.putConstraint(SpringLayout.NORTH, horizontalStrut, 0,
    SpringLayout.NORTH, panel);
119                 springLayout.putConstraint(SpringLayout.WEST, horizontalStrut, 0,
    SpringLayout.EAST, panel);
120                 frame.getContentPane().add(horizontalStrut);
121
122                 final NetworkImageViewer panel_1 = new NetworkImageViewer();
123                 panel_1.setToolTipText("Video Here");
124                 panel_1.setBackground(Color.WHITE);
```

```
125                 springLayout.putConstraint(SpringLayout.NORTH, panel_1, 0,
    SpringLayout.NORTH, panel);
126                 springLayout.putConstraint(SpringLayout.WEST, panel_1, 0,
    SpringLayout.EAST, horizontalStrut);
127                 springLayout.putConstraint(SpringLayout.SOUTH, panel_1, 480,
    SpringLayout.NORTH, panel);
128                 springLayout.putConstraint(SpringLayout.EAST, panel_1, 640,
    SpringLayout.EAST, horizontalStrut);
129                 frame.getContentPane().add(panel_1);
130                 Component verticalStrut = Box.createVerticalStrut(20);
131                 springLayout.putConstraint(SpringLayout.NORTH, verticalStrut, 0,
    SpringLayout.SOUTH, panel_1);
132                 springLayout.putConstraint(SpringLayout.WEST, verticalStrut, 0,
    SpringLayout.WEST, panel_1);
133                 springLayout.putConstraint(SpringLayout.SOUTH, verticalStrut, 10,
    SpringLayout.SOUTH, panel_1);
134                 frame.getContentPane().add(verticalStrut);
135
136                 JPanel panel_2 = new JPanel();
137                 panel_2.setBackground(Color.WHITE);
138                 springLayout.putConstraint(SpringLayout.NORTH, panel_2, 0,
    SpringLayout.SOUTH, verticalStrut);
139                 springLayout.putConstraint(SpringLayout.WEST, panel_2, 10,
    SpringLayout.EAST, panel);
140                 //springLayout.putConstraint(SpringLayout.SOUTH, panel_2, 0,
    SpringLayout.SOUTH, panel);
141                 springLayout.putConstraint(SpringLayout.EAST, panel_2, 0,
    SpringLayout.EAST, panel_1);
142                 frame.getContentPane().add(panel_2);
143                 panel_2.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
144
145                 JButton btnUp = new JButton("Forwards");
146                 btnUp.addActionListener(new ActionListener() {
147                     public void actionPerformed(ActionEvent arg0) {
148                         rover.moveManually(MoveOperation.FORWARDS);
149                     }
150                 });
151                 panel_2.add(btnUp);
152
153                 JPanel panel_2b = new JPanel();
154                 panel_2b.setBackground(Color.WHITE);
155                 springLayout.putConstraint(SpringLayout.NORTH, panel_2b, 0,
    SpringLayout.SOUTH, panel_2);
156                 springLayout.putConstraint(SpringLayout.WEST, panel_2b, 10,
    SpringLayout.EAST, panel);
157                 springLayout.putConstraint(SpringLayout.EAST, panel_2b, 0,
    SpringLayout.EAST, panel_2);
158                 frame.getContentPane().add(panel_2b);
159                 panel_2b.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
160                 /*JButton btnSelfDestruct = new JButton("Self Destruct");
161             btnSelfDestruct.setToolTipText("Boom!");
162             btnSelfDestruct.setBackground(Color.RED);
163             btnSelfDestruct.setForeground(Color.WHITE);
164             panel_2.add(btnSelfDestruct);*/
165                 JButton btnLeft = new JButton("Turn Left");
166                 btnLeft.addActionListener(new ActionListener() {
167                     public void actionPerformed(ActionEvent arg0) {
```

```
168                         rover.moveManually(MoveOperation.LEFT_TURN);
169                 }
170             });
171             panel_2b.add(btnLeft);
172
173             JButton btnStop = new JButton(" STOP ");
174             btnStop.setBackground(Color.RED);
175             btnStop.setForeground(Color.WHITE);
176             btnStop.addActionListener(new ActionListener() {
177                 public void actionPerformed(ActionEvent arg0) {
178                     /*rover.targetPosition = new Position(rover.currentPosition);
179                 rover.atTargetPosition = true;*/
180                     rover.stopRover();
181                 }
182             });
183             panel_2b.add(btnStop);
184
185             JButton btnRight = new JButton("Turn Right");
186             btnRight.addActionListener(new ActionListener() {
187                 public void actionPerformed(ActionEvent arg0) {
188                     rover.moveManually(MoveOperation.RIGHT_TURN);
189                 }
190             });
191             panel_2b.add(btnRight);
192
193             JPanel panel_2c = new JPanel();
194             springLayout.putConstraint(SpringLayout.NORTH, panel_2c, 0,
    SpringLayout.SOUTH, panel_2b);
195             springLayout.putConstraint(SpringLayout.WEST, panel_2c, 10,
    SpringLayout.EAST, panel);
196             springLayout.putConstraint(SpringLayout.SOUTH, panel_2c, 33,
    SpringLayout.SOUTH, panel_2b);
197             springLayout.putConstraint(SpringLayout.EAST, panel_2c, 0,
    SpringLayout.EAST, panel_2);
198             panel_2c.setBackground(Color.WHITE);
199             frame.getContentPane().add(panel_2c);
200             panel_2c.setLayout(new FlowLayout(FlowLayout.CENTER, 5, 5));
201
202             JButton btnDown = new JButton("Backwards");
203             btnDown.addActionListener(new ActionListener() {
204                 public void actionPerformed(ActionEvent arg0) {
205                     rover.moveManually(MoveOperation.BACKWARDS);
206                 }
207             });
208             panel_2c.add(btnDown);
209             //Warn the user if downloaded maps cannot be found
210             if((!Files.exists(Paths.get("./mapcache/17"))) &&
    (!Files.exists(Paths.get(System.getProperty("user.home") + "/mapcache/17")))) {
211                 JOptionPane.showMessageDialog(frame, "Could not find downloaded
    maps. Mapping will be unavailable.");
212             }
213             final Console console = new Console();
214             rover = new NetworkRover();
215             rover.attachListener(console);
216             module.attachListener(console);
217             map_2 = new CustomMap(rover);
```

```
218                 springLayout.putConstraint(SpringLayout.NORTH, map_2, 10,
    SpringLayout.NORTH, frame.getContentPane());
219                 springLayout.putConstraint(SpringLayout.WEST, map_2, 10,
    SpringLayout.EAST, panel_1);
220                 springLayout.putConstraint(SpringLayout.SOUTH, map_2, 300,
    SpringLayout.NORTH, frame.getContentPane());
221                 springLayout.putConstraint(SpringLayout.EAST, map_2, -10,
    SpringLayout.EAST, frame.getContentPane());
222                 frame.getContentPane().add( map_2);
223                 module.attachListener(map_2);
224                 Runnable roverUpdater = new Runnable() {
225                     public void run() {
226                         rover.update();
227                     }
228                 };
229
230
231                 //Schedule automated updating of the simulated rover.
232                 updater = Executors.newScheduledThreadPool(1);
233                 updater.scheduleAtFixedRate(roverUpdater, 0, 1000,
    TimeUnit.MILLISECONDS);
234
235
236                 Runnable moduleUpdater = new Runnable() {
237                     public void run() {
238                         module.update();
239                     }
240                 };
241
242
243                 //Schedule automated updating of the simulated rover.
244                 updater1 = Executors.newScheduledThreadPool(4);
245                 updater1.scheduleAtFixedRate(moduleUpdater, 0, 2000,
    TimeUnit.MILLISECONDS);
246
247                 Runnable antennaUpdater = new Runnable() {
248                     public void run() {
249                         ant.updateAntennaPosition(comms);
250                     }
251                 };
252
253
254                 //Schedule automated updating of the simulated rover.
255                 updater2 = Executors.newScheduledThreadPool(1);
256                 updater2.scheduleAtFixedRate(antennaUpdater, 0, 1000,
    TimeUnit.MILLISECONDS);
257
258                 JPanel panel_3 = new JPanel();
259                 //panel_3.setBackground(Color.WHITE);
260                 springLayout.putConstraint(SpringLayout.NORTH, panel_3, 10,
    SpringLayout.SOUTH, map_2);
261                 springLayout.putConstraint(SpringLayout.WEST, panel_3, 10,
    SpringLayout.EAST, panel_1);
262                 springLayout.putConstraint(SpringLayout.SOUTH, panel_3, -10,
    SpringLayout.SOUTH, frame.getContentPane());
263                 springLayout.putConstraint(SpringLayout.EAST, panel_3, -10,
    SpringLayout.EAST, frame.getContentPane());
```

```
264                 frame.getContentPane().add(panel_3);
265
266                 SpringLayout sl_panel_3 = new SpringLayout();
267                 panel_3.setLayout(sl_panel_3);
268                 JLabel lblRawDataHere = new JLabel("Raw Data Here");
269                 sl_panel_3.putConstraint(SpringLayout.NORTH, lblRawDataHere, 11,
    SpringLayout.NORTH, panel_3);
270                 sl_panel_3.putConstraint(SpringLayout.WEST, lblRawDataHere, 110,
    SpringLayout.WEST, panel_3);
271                 panel_3.add(lblRawDataHere);
272
273                 JPanel panel_4 = new JPanel();
274                 sl_panel_3.putConstraint(SpringLayout.NORTH, panel_4, 319,
    SpringLayout.NORTH, panel_3);
275                 sl_panel_3.putConstraint(SpringLayout.WEST, panel_4, 0,
    SpringLayout.WEST, panel_3);
276                 sl_panel_3.putConstraint(SpringLayout.SOUTH, panel_4, 381,
    SpringLayout.NORTH, panel_3);
277                 sl_panel_3.putConstraint(SpringLayout.EAST, panel_4, 294,
    SpringLayout.WEST, panel_3);
278                 panel_3.add(panel_4);
279
280                 JCheckBox chckbxData = new JCheckBox("Data 1");
281                 panel_4.add(chckbxData);
282
283                 JCheckBox chckbxData_1 = new JCheckBox("Data 2");
284                 panel_4.add(chckbxData_1);
285
286                 JCheckBox chckbxData_2 = new JCheckBox("Data 3");
287                 panel_4.add(chckbxData_2);
288
289                 JCheckBox chckbxData_3 = new JCheckBox("Data 4");
290                 panel_4.add(chckbxData_3);
291
292                 JButton btnConvertIntoDem = new JButton("Open Large Map");
293                 btnConvertIntoDem.addActionListener(new ActionListener() {
294                     @Override
295                     public void actionPerformed(ActionEvent arg0) {
296                         m = new LargeMap(rover,module,new
    Position(map_2.getPosition()),map_2.getZoom());
297                     }
298
299                 });
300                 panel_4.add(btnConvertIntoDem);
301
302                 JButton btnCreateGraphs = new JButton("Create Graphs");
303                 panel_4.add(btnCreateGraphs);
304
305                 JPanel panel_2d = new JPanel();
306                 springLayout.putConstraint(SpringLayout.NORTH, panel_2d, 0,
    SpringLayout.SOUTH, panel_2c);
307                 springLayout.putConstraint(SpringLayout.WEST, panel_2d, 10,
    SpringLayout.EAST, panel);
308                 springLayout.putConstraint(SpringLayout.SOUTH, panel_2d, 33,
    SpringLayout.SOUTH, panel_2c);
309
310                 JScrollPane scrollPane = new JScrollPane(console,
```

```
311                         JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
312                         JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
313             sl_panel_3.putConstraint(SpringLayout.NORTH, scrollPane, 0,
    SpringLayout.SOUTH, lblRawDataHere);
314             sl_panel_3.putConstraint(SpringLayout.WEST, scrollPane, 0,
    SpringLayout.WEST, panel_4);
315             sl_panel_3.putConstraint(SpringLayout.SOUTH, scrollPane, 0,
    SpringLayout.NORTH, panel_4);
316             sl_panel_3.putConstraint(SpringLayout.EAST, scrollPane, 0,
    SpringLayout.EAST, panel_3);
317             panel_3.add(scrollPane);
318             //panel_2d.setBackground(Color.WHITE);
319             frame.getContentPane().add(panel_2d);
320
321             JLabel lblDebugControls = new JLabel("Debug Controls:");
322             panel_2d.add(lblDebugControls);
323
324             JButton btnStart = new JButton("Start");
325             panel_2d.add(btnStart);
326             btnStart.addActionListener(new ActionListener() {
327                 @Override
328                 public void actionPerformed(ActionEvent arg0) {
329                     console.startDebug();
330                 }
331             });
332
333             JButton btnPause = new JButton("Pause");
334             panel_2d.add(btnPause);
335             btnPause.addActionListener(new ActionListener() {
336                 @Override
337                 public void actionPerformed(ActionEvent arg0) {
338                     console.pauseDebug();
339                 }
340             });
341
342             JLabel lblVideoControls = new JLabel("Video Controls:");
343             panel_2d.add(lblVideoControls);
344
345             JButton btnStartRecord = new JButton("Start Record");
346             panel_2d.add(btnStartRecord);
347
348             JButton btnStopRecord = new JButton("Stop Record");
349             panel_2d.add(btnStopRecord);
350
351             //final NetworkSender ns = new NetworkSender("127.0.0.1"); //For testing only,
    change to real IP later
352
353             JPanel panel_2e = new JPanel();
354             springLayout.putConstraint(SpringLayout.NORTH, panel_2e, 0,
    SpringLayout.SOUTH, panel_2d);
355             springLayout.putConstraint(SpringLayout.WEST, panel_2e, 10,
    SpringLayout.EAST, panel);
356             //springLayout.putConstraint(SpringLayout.EAST, panel_2e, 10,
    SpringLayout.WEST, panel_3);
357
358             springLayout.putConstraint(SpringLayout.SOUTH, panel_2e, 33,
    SpringLayout.SOUTH, panel_2d);
```

```
359                 //panel_2e.setBackground(Color.WHITE);
360                 frame.getContentPane().add(panel_2e);
361                 JLabel lblNetwork = new JLabel("Network:");
362                 panel_2e.add(lblNetwork);
363 
364                 JButton btnNetworkStart = new JButton("Start");
365                 panel_2e.add(btnNetworkStart);
366                 btnNetworkStart.addActionListener(new ActionListener() {
367                     @Override
368                     public void actionPerformed(ActionEvent e) {
369                         rover.begin(settings.get("rover.ip"));
370                         panel_1.startDisplay();
371                     }
372                 });
373                 JButton btnNetworkTest = new JButton("Test");
374                 panel_2e.add(btnNetworkTest);
375                 btnNetworkTest.addActionListener(new ActionListener() {
376                     @Override
377                     public void actionPerformed(ActionEvent e) {
378                         NetworkStatus status = rover.testNetwork();
379                         long startTime = System.currentTimeMillis();
380                         if(status == NetworkStatus.OK) {
381                             JOptionPane.showMessageDialog(null,"Network
    OK!\nLatency: " + (System.currentTimeMillis() - startTime) + "ms");
382                         } else {
383                             JOptionPane.showMessageDialog(null,"Network error: " +
    status.toString());
384                         }
385                     }
386                 });
387                 JButton btnNetworkStop = new JButton("Stop");
388                 panel_2e.add(btnNetworkStop);
389                 btnNetworkStop.addActionListener(new ActionListener() {
390                     @Override
391                     public void actionPerformed(ActionEvent e) {
392                         rover.disconnect();
393                         panel_1.stopDisplay();
394                     }
395                 });
396 
397                 JLabel lblMapControls = new JLabel("Map: ");
398                 panel_2e.add(lblMapControls);
399                 JButton btnMapGoto = new JButton("Go To");
400                 panel_2e.add(btnMapGoto);
401                 btnMapGoto.addActionListener(new ActionListener() {
402                     @Override
403                     public void actionPerformed(ActionEvent arg0) {
404                         GoToDialog dialog = new GoToDialog(frame);
405                         GoToDialogResult result =
    dialog.show(map_2.getPosition().getLat(),map_2.getPosition().getLon());
406                         if(result.success) {
407                             switch(result.selectedOption) {
408                             case GOTO_ROVER:
409 
    map_2.setDisplayPositionByLatLon(rover.currentPosition.getLat(),
410                                                 rover.currentPosition.getLon(),
411                                                 map_2.getZoom());
```

```
412                                     break;
413                                 case GOTO_MODULE:
414                                     //TODO
415                                     break;
416                                 case GOTO_LATLONG:
417
    map_2.setDisplayPositionByLatLon(result.enteredPosition.getLat(),
418                                                     result.enteredPosition.getLon(),
419                                                     map_2.getZoom());
420                                     break;
421                                 }
422                             }
423                         }
424                 });
425
426                 JLabel lblArduino = new JLabel("Arduino: ");
427                 panel_2e.add(lblArduino);
428                 JButton btnStartBS = new JButton("Start");
429                 panel_2e.add(btnStartBS);
430                 btnStartBS.addActionListener(new ActionListener() {
431
432                     @Override
433                     public void actionPerformed(ActionEvent arg0) {
434                         comms.startCommunications(settings.get("serial.port"));
435                         module.startCommunications(comms);
436                         module.attachListener(ant);
437                         rover.attachListener(ant);
438                     }
439                 });
440
441                 JPanel panel_2f = new JPanel();
442                 springLayout.putConstraint(SpringLayout.NORTH, panel_2f, 0,
    SpringLayout.SOUTH, panel_2e);
443                 springLayout.putConstraint(SpringLayout.WEST, panel_2f, 10,
    SpringLayout.EAST, panel);
444                 springLayout.putConstraint(SpringLayout.SOUTH, panel_2f, 33,
    SpringLayout.SOUTH, panel_2e);
445                 frame.getContentPane().add(panel_2f);
446                 JButton btnSettings = new JButton("Settings");
447                 btnSettings.addActionListener(new ActionListener() {
448                     @Override
449                     public void actionPerformed(ActionEvent arg0) {
450                         @SuppressWarnings("unused")
451                         SettingsWindow s = new SettingsWindow();
452                     }
453                 });
454                 panel_2f.add(btnSettings);
455
456                 JLabel lblDeployment = new JLabel("Deployment:");
457                 panel_2f.add(lblDeployment);
458                 JButton btnDeploy = new JButton("Deploy");
459                 btnDeploy.addActionListener(new ActionListener() {
460
461                     @Override
462                     public void actionPerformed(ActionEvent arg0) {
463                         if(JOptionPane.showConfirmDialog(frame,
464                                 "Are you sure you want to deploy the rover?",
```

```
465                                     "Deployment",
466                                     JOptionPane.YES_NO_OPTION) ==
    JOptionPane.YES_OPTION) {
467                                 module.deployRover();
468                             }
469
470                     }
471                 });
472                 panel_2f.add(btnDeploy);
473                 JButton btnAbort = new JButton("Abort");
474                 btnAbort.setBackground(Color.RED);
475                 btnAbort.setForeground(Color.WHITE);
476                 btnAbort.addActionListener(new ActionListener() {
477
478                     @Override
479                     public void actionPerformed(ActionEvent arg0) {
480                             module.terminateDeployment();
481                     }
482
483
484                 });
485                 panel_2f.add(btnAbort);
486
487                 JLabel lblEEPROM = new JLabel("EEPROM:");
488                 panel_2f.add(lblEEPROM);
489                 JButton btnFormatEEPROM = new JButton("Format");
490                 btnFormatEEPROM.addActionListener(new ActionListener() {
491
492                     @Override
493                     public void actionPerformed(ActionEvent arg0) {
494                         if(JOptionPane.showConfirmDialog(frame,
495                                 "Are you sure you want to format the rover's
    EEPROM?",
496                                 "EEPROM Operation",
497                                 JOptionPane.YES_NO_OPTION) ==
    JOptionPane.YES_OPTION) {
498                             rover.formatEEPROM();
499                         }
500
501                     }
502                 });
503                 panel_2f.add(btnFormatEEPROM);
504
505                 //panel_1.updateImage();
506                 frame.toFront();
507                 //frame.setExtendedState(JFrame.MAXIMIZED_BOTH);
508                 d = new DataLogger();
509                 d.start();
510                 rover.attachListener(d);
511                 module.attachListener(d);
512         }
513     }
```

6.2 AntennaManager.java

```
1   package gui;
2
3   import gui.LandingModule.DeploymentStatus;
```

```
4
5   public class AntennaManager implements RoverUpdateListener,
6           LandingModuleListener {
7       private boolean havePositionFix = false;
8       private Position currentTrackingPosition = null;
9       private boolean trackingRover = false;
10      private double currentTrackingAltitude = 0;
11      private SettingsStore settings;
12      public AntennaManager() {
13          settings = new SettingsStore();
14      }
15
16      @Override
17      public void positionUpdated(Position newPosition, LandingModule m) {
18          if(!trackingRover) {
19              if(m.gpsAvailable) {
20                  currentTrackingPosition = newPosition;
21                  currentTrackingAltitude = (int) m.gpsAltitude;
22                  havePositionFix = true;
23              }
24          }
25
26      }
27
28      @Override
29      public void dataUpdated(TPData currentData, LandingModule m) {
30          // TODO Auto-generated method stub
31
32      }
33
34      @Override
35      public void statusUpdated(DeploymentStatus deployed, boolean connected,
36              LandingModule m) {
37          if(deployed == DeploymentStatus.DEPLOYED ) {
38              trackingRover = true;
39          }
40
41      }
42
43      @Override
44      public void positionUpdated(Position newPosition, Position targetPosition,
45              boolean atTargetPosition, Rover r) {
46          if(trackingRover) {
47              if(newPosition.getLat() != 0) {
48                  currentTrackingPosition = newPosition;
49                  havePositionFix = true;
50              }
51          }
52
53      }
54
55      @Override
56      public void dataUpdated(TPData data, Rover r) {
57          // TODO Auto-generated method stub
58
59      }
60
```

```
61      @Override
62      public void messageRecieved(String message, Rover r) {
63          // TODO Auto-generated method stub
64
65      }
66
67      @Override
68      public void errorThrown(String message, Rover r) {
69          // TODO Auto-generated method stub
70
71      }
72
73      public void updateAntennaPosition(BaseStationCommunications b) {
74          //System.err.println("HI");
75          if(havePositionFix) {
76
77              Position home = new Position(settings.getDouble("home.lat"),
78                      settings.getDouble("home.long"));
79              double homeAlt = settings.getDouble("home.alt");
80              double horizontalDistance =
    home.getDistanceTo(currentTrackingPosition);
81              double horizontalAngle = currentTrackingPosition.getBearingTo(home);
82              double verticalAngle =
    Math.toDegrees(Math.atan((currentTrackingAltitude - homeAlt) / horizontalDistance));
83              System.err.println(horizontalAngle +  " " + verticalAngle);
84              b.setAntennaTilt((int)verticalAngle, (int)horizontalAngle);
85          }
86      }
87  }
```

6.3 BaseStationCommunications.java

```
1   package gui;
2
3   import java.io.BufferedReader;
4   import java.io.BufferedWriter;
5   import java.io.IOException;
6   import java.io.InputStreamReader;
7   import java.io.OutputStreamWriter;
8
9   /*
10  * This class is designed to provide an interface to the Arduino on the base station
11  * which is responsible for radio communications and antenna positioning.
12  */
13  public class BaseStationCommunications {
14
15      private boolean busy = false;
16      Process p;
17      BufferedWriter out;
18      BufferedReader in;
19      public BaseStationCommunications() {
20          // TODO Auto-generated constructor stub
21      }
22
23      //True = OK
24      //False = Error
25      //Open the serial port
26      public boolean startCommunications(String port) {
```

```
27              try {
28                      SettingsStore settings = new SettingsStore();
29                      p = Runtime.getRuntime().exec("SerialWrapper.exe");
30                      in = new BufferedReader( new InputStreamReader(p.getInputStream())
    );
31              out = new BufferedWriter( new OutputStreamWriter(p.getOutputStream()) );
32              out.write(settings.get("serial.port") + "\n");
33              out.flush();
34                      busy = false;
35
36              } catch (Exception e) {
37                      e.printStackTrace();
38              }
39              //serialPortWriter.println("C S");
40              return true;
41      }
42
43      private boolean waitForFreePort() {
44              long startWaitTime = System.currentTimeMillis();
45              while(busy) {
46                      if((System.currentTimeMillis() - startWaitTime) > 5000) {
47                              return false;
48                      }
49              }
50              return true;
51      }
52
53      //Close the serial port
54      public void stopCommunications() {
55              waitForFreePort();
56              //serialPortWriter.println("C E");
57              busy = false;
58              try {
59                      out.write("Q\n");
60                      out.flush();
61              } catch (IOException e1) {
62                      // TODO Auto-generated catch block
63                      e1.printStackTrace();
64              }
65              try { Thread.sleep(100); } catch (InterruptedException e) { }
66              p.destroy();
67              p = null;
68
69      }
70      //True = OK
71      //False = Error
72      //Sets WiFi antenna target tilt (up/down) angle and pan (left/right) angle
73      public boolean setAntennaTilt(int tilt, int pan) {
74              if(!waitForFreePort())
75                      return false;
76              busy = true;
77              try {
78                      out.write("A " + pan + " " + tilt  + "\n");
79                      out.flush();
80              } catch (IOException e) {
81                      // TODO Auto-generated catch block
82                      e.printStackTrace();
```

```
83            }
84
85            busy = false;
86            return true;
87      }
88
89      public String getAvailableRadioData() {
90            if(!waitForFreePort())
91                  return "";
92            busy = true;
93            try {
94            out.write("R\n");
95            out.flush();
96            } catch (IOException e) {
97
98            }
99            String s = "", total = "";
100               while(true) {
101                     try {
102                           s = in.readLine();
103                           if(s.contains("-----END-----")) break;
104                           total += s;
105                           //System.err.println(s);
106                     } catch (IOException e) {
107                           s = "";
108                           System.err.println("--ERR IN gARD---");
109                           e.printStackTrace();
110                     }
111               }
112               busy = false;
113               return total;
114         }
115
116         public boolean sendRadio(String data) {
117               if(!waitForFreePort())
118                     return false;
119               busy = true;
120               try {
121                     out.write("W " + data + "\n");
122                     out.flush();
123               } catch (IOException e) {
124                     // TODO Auto-generated catch block
125                     e.printStackTrace();
126               }
127               busy = false;
128               return true;
129         }
130
131
132   }
```

6.4 Console.java

```
1   package gui;
2
3   import gui.LandingModule.DeploymentStatus;
4
5   import java.awt.Color;
```

```
6  import java.awt.Font;
7
8  import javax.swing.JTextArea;
9
10 public class Console extends JTextArea implements RoverUpdateListener,
   LandingModuleListener {
11      private static final long serialVersionUID = 4989941948035851333L;
12      public boolean paused = false;
13      Console() {
14          super();
15          setLineWrap(true);
16          setColumns(25);
17          setFont(new Font("Lucida Console", Font.PLAIN, 12));
18          setForeground(Color.GREEN);
19          setBackground(Color.BLACK);
20          setEditable(false);
21      }
22
23      public void startDebug() {
24          paused = false;
25      }
26
27      public void pauseDebug() {
28          paused = true;
29      }
30
31      @Override
32      public void positionUpdated(Position newPosition, Position targetPosition, boolean
   atTargetPosition,
33              Rover r) {
34          if(!paused) {
35              append("D1|Location: " + newPosition.toString() + "\n");
36              setCaretPosition(getDocument().getLength());
37          };
38      }
39      @Override
40      public void dataUpdated(TPData data, Rover r) {
41          if(!paused) {
42              append("D1|TP: " + data.toString() + "\n");
43              setCaretPosition(getDocument().getLength());
44          };
45      }
46      @Override
47      public void messageRecieved(String message, Rover r) {
48          // TODO Auto-generated method stub
49
50      }
51      @Override
52      public void errorThrown(String message, Rover r) {
53          if(!paused) {
54              append("ERROR: " + message + "\n");
55              setCaretPosition(getDocument().getLength());
56          };
57      }
58
59      @Override
60      public void positionUpdated(Position newPosition, LandingModule m) {
```

```
61              if(!paused) {
62                      append("LM|Location: " + newPosition.toString() + "\n");
63                      setCaretPosition(getDocument().getLength());
64              };
65
66      }
67
68      @Override
69      public void dataUpdated(TPData currentData, LandingModule m) {
70              if(!paused) {
71                      append("LM|TP: " + currentData.toString() + "\n");
72                      setCaretPosition(getDocument().getLength());
73              };
74
75      }
76
77      @Override
78      public void statusUpdated(DeploymentStatus deployed, boolean connected,
79                      LandingModule m) {
80              // TODO Auto-generated method stub
81
82      }
83  }
```

6.5 CustomMap.java

```
1   package gui;
2
3   import gui.LandingModule.DeploymentStatus;
4
5   import java.awt.Color;
6   import java.nio.file.Files;
7   import java.nio.file.Paths;
8
9   import org.openstreetmap.gui.jmapviewer.JMapViewer;
10  import org.openstreetmap.gui.jmapviewer.MapMarkerDot;
11  import org.openstreetmap.gui.jmapviewer.tilesources.OfflineOsmTileSource;
12
13  /*
14   * This is a customised version of JMapViewer that has built-in handling of
15   * rover position display and updating, as well as registration of the click handler.
16   *
17   */
18
19  public class CustomMap extends JMapViewer implements RoverUpdateListener,
    LandingModuleListener {
20      private static final long serialVersionUID = 4892531410795284424L;
21      Rover rover;
22      MapMarkerDot actualPos, targetPos, modulePos;
23      SettingsStore settings = new SettingsStore();
24      public CustomMap(Rover r) {
25              super();
26              this.rover = r;
27              rover.attachListener(this);
28              addMouseListener(new MapClickHandler(rover));
29              if(Files.exists(Paths.get("./mapcache/17"))) {
30                      setTileSource(new OfflineOsmTileSource("file:///" +
    System.getProperty("user.dir").replace("\\","/") + "/mapcache/",14,18));
```

```
31              } else {
32                      setTileSource(new OfflineOsmTileSource("file:///" +
System.getProperty("user.home").replace("\\","/") + "/mapcache/",14,18));
33
34              }
35
setDisplayPositionByLatLon(settings.getDouble("home.lat"),settings.getDouble("home.long
"), 16);
36      }
37      @Override
38      public void positionUpdated(Position newPosition, Position targetPosition, boolean
atTargetPosition,
39                      Rover r) {
40              try {
41                      removeMapMarker(actualPos);
42                      removeMapMarker(targetPos);
43              } finally {
44
45              };
46       actualPos = new MapMarkerDot(Color.green
47                      , newPosition.getLat()
48                      , newPosition.getLon());
49       addMapMarker(actualPos);
50       if(!atTargetPosition) {
51              targetPos = new MapMarkerDot(Color.red
52                              , targetPosition.getLat()
53                              , targetPosition.getLon());
54              addMapMarker(targetPos);
55       };
56
57      }
58
59      // --- ROVER ---
60
61      @Override
62      public void dataUpdated(TPData data, Rover r) {
63              // Not used at present
64
65      }
66      @Override
67      public void messageRecieved(String message, Rover r) {
68              // Not used at present
69
70      }
71      @Override
72      public void errorThrown(String message, Rover r) {
73              // TODO Auto-generated method stub
74
75      }
76      public void shutDown() {
77              rover.removeListener(this);
78      }
79      // --- LANDING MODULE ---
80      @Override
81      public void positionUpdated(Position newPosition, LandingModule m) {
82              try {
83                      removeMapMarker(modulePos);
```

```
84              } finally {
85              modulePos = new MapMarkerDot(Color.yellow
86                              , newPosition.getLat()
87                              , newPosition.getLon());
88              addMapMarker(modulePos);
89              };
90
91
92      }
93      @Override
94      public void dataUpdated(TPData currentData, LandingModule m) {
95              // TODO Auto-generated method stub
96
97      }
98      @Override
99      public void statusUpdated(DeploymentStatus deployed, boolean connected,
100                             LandingModule m) {
101                     // TODO Auto-generated method stub
102
103             }
104     }
```

6.6 DataLogger.java

```
1   package gui;
2
3   import java.io.FileOutputStream;
4   import java.io.IOException;
5   import java.io.PrintWriter;
6   import java.nio.file.Files;
7   import java.nio.file.Paths;
8   import java.text.DateFormat;
9   import java.text.SimpleDateFormat;
10  import java.util.Date;
11
12
13  import gui.LandingModule.DeploymentStatus;
14
15  public class DataLogger implements RoverUpdateListener, LandingModuleListener {
16      Position currentRoverPosition = new Position(0,0), currentLandingModulePosition =
    new Position(0,0);
17      TPData currentRoverTP = new TPData(0,0), currentLandingModuleTP = new
    TPData(0,0);
18      String filename;
19      boolean logging = false;
20      public DataLogger() {
21              currentLandingModulePosition = new Position(0, 0);
22              currentRoverPosition = new Position(0, 0);
23
24      }
25
26      public boolean start() {
27              if(!Files.exists(Paths.get("./data/"))) {
28                      try {
29                              Files.createDirectory(Paths.get("./data/"));
30                      } catch (IOException e) {
31                              return false;
32                      }
```

```
33
34              }
35              DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd HHmmss");
36              Date date = new Date();
37              filename = "./data/log-" + dateFormat.format(date) + ".csv";
38              logging = true;
39              putLine("Time,Rover Latitude, Rover Longitude, Rover Temperature, Rover
    Pressure,"
40                              + "Landing Module Latitude, Landing Module Longitude,
    Landing Module Temperature, Landing Module Pressure");
41              return true;
42      }
43
44      private void putLine(String string) {
45              FileOutputStream outputStream;
46              PrintWriter fileWriter;
47              try {
48                      outputStream = new FileOutputStream(filename,true); //Open file for
    appending
49                      fileWriter = new PrintWriter(outputStream);
50                      fileWriter.println(string);
51                      fileWriter.close();
52                      outputStream.close();
53                      outputStream = null;
54                      fileWriter = null;
55              } catch(Exception e) {
56                      e.printStackTrace();
57              }
58      }
59
60      private void addDataLine() {
61              if(logging) {
62                      String dataLine = "";
63                      DateFormat dateFormat = new SimpleDateFormat("HH:mm:ss");
64                      Date date = new Date();
65                      dataLine += dateFormat.format(date);
66                      dataLine += ",";
67                      dataLine += currentRoverPosition.getLat();
68                      dataLine += ",";
69                      dataLine += currentRoverPosition.getLon();
70                      dataLine += ",";
71                      dataLine += currentRoverTP.temperature;
72                      dataLine += ",";
73                      dataLine += currentRoverTP.pressure;
74                      dataLine += ",";
75                      dataLine += currentLandingModulePosition.getLat();
76                      dataLine += ",";
77                      dataLine += currentLandingModulePosition.getLon();
78                      dataLine += ",";
79                      dataLine += currentLandingModuleTP.temperature;
80                      dataLine += ",";
81                      dataLine += currentLandingModuleTP.pressure;
82                      putLine(dataLine);
83              };
84      }
85
86      @Override
```

```
87      public void positionUpdated(Position newPosition, LandingModule m) {
88          currentLandingModulePosition = newPosition;
89          addDataLine();
90      }
91
92      @Override
93      public void dataUpdated(TPData currentData, LandingModule m) {
94          currentLandingModuleTP = currentData;
95          addDataLine();
96      }
97
98      @Override
99      public void statusUpdated(DeploymentStatus deployed, boolean connected,
100                 LandingModule m) {
101             // TODO Auto-generated method stub
102
103         }
104
105         @Override
106         public void positionUpdated(Position newPosition, Position targetPosition,
107                 boolean atTargetPosition, Rover r) {
108             currentRoverPosition = newPosition;
109             addDataLine();
110         }
111
112         @Override
113         public void dataUpdated(TPData data, Rover r) {
114             currentRoverTP = data;
115             addDataLine();
116         }
117
118         @Override
119         public void messageRecieved(String message, Rover r) {
120             // TODO Auto-generated method stub
121
122         }
123
124         @Override
125         public void errorThrown(String message, Rover r) {
126             // TODO Auto-generated method stub
127
128         }
129
130     }
```

6.7 GoToDialog.java

```
1   package gui;
2
3   import javax.swing.ButtonGroup;
4   import javax.swing.JComponent;
5   import javax.swing.JFrame;
6   import javax.swing.JLabel;
7   import javax.swing.JOptionPane;
8   import javax.swing.JRadioButton;
9   import javax.swing.JTextField;
10
11  public class GoToDialog {
```

```
12      JFrame parent;
13      private SettingsStore settings;
14      public GoToDialog(JFrame dialogParent) {
15            parent = dialogParent;
16            settings = new SettingsStore();
17      }
18      public GoToDialogResult show(double defaultLat, double defaultLon) {
19            JRadioButton toRoverPosition = new JRadioButton("Go to Rover");
20            toRoverPosition.setSelected(true); //Default option
21            JRadioButton toLandingModulePosition = new JRadioButton("Go to Landing
Module");
22            JRadioButton toHomeLocation = new JRadioButton("Go to Home");
23            JRadioButton toLatLongPair = new JRadioButton("Go to
Latitude/Longitude:");
24
25            ButtonGroup gotoOptions = new ButtonGroup();
26            gotoOptions.add(toRoverPosition);
27            gotoOptions.add(toLandingModulePosition);
28            gotoOptions.add(toHomeLocation);
29            gotoOptions.add(toLatLongPair);
30
31            JTextField latitudeField = new JTextField(Double.toString(defaultLat));
32            JTextField longitudeField = new JTextField(Double.toString(defaultLon));
33
34            final JComponent[] inputs = new JComponent[] {
35                  toRoverPosition,
36                  toLandingModulePosition,
37                  toHomeLocation,
38                  toLatLongPair,
39                  new JLabel("Latitude"),
40                  latitudeField,
41                  new JLabel("Longitude"),
42                  longitudeField
43            };
44            GoToDialogResult goToWhere;
45            int dialogResult = JOptionPane.showConfirmDialog(parent, inputs,"Go to
where?",JOptionPane.OK_CANCEL_OPTION);
46            if(dialogResult == JOptionPane.CANCEL_OPTION) {
47                  goToWhere = new GoToDialogResult(false, null, null);
48            } else {
49                  if(toRoverPosition.isSelected()) {
50                        goToWhere = new GoToDialogResult(true,
GoToDialogResult.Selection.GOTO_ROVER, null);
51                  } else if(toLandingModulePosition.isSelected()) {
52                        goToWhere = new GoToDialogResult(true,
GoToDialogResult.Selection.GOTO_MODULE, null);
53                  } else if(toHomeLocation.isSelected()) {
54                        goToWhere = new GoToDialogResult(true,
GoToDialogResult.Selection.GOTO_LATLONG,
55                                    new
Position(settings.getDouble("home.lat"),settings.getDouble("home.long")));
56                  } else if(toLatLongPair.isSelected()) {
57                        double latitude, longitude;
58                        try {
59                              latitude = Double.parseDouble(latitudeField.getText());
60                              longitude = Double.parseDouble(longitudeField.getText());
61                              goToWhere = new GoToDialogResult(true,
```

```
                                        GoToDialogResult.Selection.GOTO_LATLONG,
                                        new Position(latitude, longitude));
                        } catch (NumberFormatException e) {
                                JOptionPane.showMessageDialog(parent, "Invalid
latitude/longitude","Error",JOptionPane.WARNING_MESSAGE);
                                goToWhere = new GoToDialogResult(false, null, null);
                        }
                } else {
                        goToWhere = new GoToDialogResult(false, null, null);
                }
        }
        return goToWhere;
    }
}
```

6.8 GoToDialogResult.java

```
package gui;

public class GoToDialogResult {
    enum Selection {
        GOTO_ROVER,
        GOTO_MODULE,
        GOTO_LATLONG
    }
    boolean success;
    Selection selectedOption;
    Position enteredPosition;
    public GoToDialogResult(boolean success, Selection selectedOption,
            Position enteredPosition) {
        this.success = success;
        this.selectedOption = selectedOption;
        this.enteredPosition = enteredPosition;
    }
}
```

6.9 LandingModule.java

```
package gui;

import java.util.Vector;
import java.util.concurrent.Executors;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.TimeUnit;

/*
 * Radio sentence format
 * STARTdataEND
 * data: temp,pressure,gpsAvailable,lat,long,alt,deployed (comma separated)
 * e.g. sentence: B175,100000,1,51.487556,-0.2381855,100,1E
 * temp: temperature in degrees C * 10
 * pressure: pressure in Pa
 * gpsAvailable: 1 if GPS fix available, 0 otherwise
 * lat: GPS latitude
 * long: GPS longitude
 * alt: GPS (not barometric) altitude
 * state: 7 if rover deployed, ? if not
 */
```

```
21
22 public class LandingModule {
23     public enum DeploymentStatus {
24         LOADED, DEPLOYED, UNKNOWN
25     }
26
27     private ScheduledExecutorService updater;
28     Vector<LandingModuleListener> listeners = new Vector<LandingModuleListener>();
29     public Position currentPosition = new Position(0, 0);
30     protected TPData currentData = new TPData(0, 0);
31     public DeploymentStatus roverDeployed = DeploymentStatus.LOADED;
32     public boolean connected = false, running = false, gpsAvailable = false;
33     String radioBuffer = "";
34     private BaseStationCommunications baseStation;
35     public double gpsAltitude = 0;
36     public int status = -1;
37
38     public void attachListener(LandingModuleListener l) {
39         listeners.add(l);
40     };
41
42     public void removeListener(LandingModuleListener l) {
43         if (listeners.contains(l)) {
44             listeners.remove(l);
45         }
46         ;
47     };
48
49     public void deployRover() {
50         Runnable radioSender = new Runnable() {
51             public void run() {
52                 if ((status == 6) || (status == 7)) {
53                     updater.shutdown();
54                 }
55                 baseStation.sendRadio("DEP");
56             }
57         };
58
59         // Keep going until accepted
60         updater = Executors.newScheduledThreadPool(1);
61         updater.scheduleAtFixedRate(radioSender, 0, 1000, TimeUnit.MILLISECONDS);
62     }
63
64     public void terminateDeployment() {
65         Runnable radioSender = new Runnable() {
66             public void run() {
67                 if (status == 8) {
68                     updater.shutdown();
69                 }
70                 baseStation.sendRadio("END");
71             }
72         };
73
74         // Keep going until accepted
75         updater = Executors.newScheduledThreadPool(1);
76         updater.scheduleAtFixedRate(radioSender, 0, 1000, TimeUnit.MILLISECONDS);
77     }
```

```
78
79      public void startCommunications(BaseStationCommunications baseArduinoComms) {
80            baseStation = baseArduinoComms;
81            running = true;
82      }
83
84      public void stopCommunications() {
85            running = false;
86      }
87
88      public void update() {
89
90            if (running) {
91                  boolean packetRecieved = false;
92                  // System.err.println("...");
93                  String newData = baseStation.getAvailableRadioData();
94                  radioBuffer += newData;
95                  System.err.println(radioBuffer);
96                  try {
97                        while (true) {
98                              int startOfSentence = radioBuffer.indexOf("START");
99                              int endOfSentence = radioBuffer.indexOf("END",
   startOfSentence + 5);
100                             if (endOfSentence < 0)
101                                   break;
102                             if (startOfSentence < 0)
103                                   break; // No full packet remaining, stop
104
105                             String sentence = radioBuffer.substring(startOfSentence +
   5, endOfSentence);
106                             System.out.println(sentence);
107                             radioBuffer = radioBuffer.substring(endOfSentence + 3);
108                             String splitSentence[] = sentence.split(",");
109                             // if(splitSentence.length < 7) continue;
110                             currentData.temperature =
   Double.parseDouble(splitSentence[0]) / 100;
111                             currentData.pressure =
   Double.parseDouble(splitSentence[1]);
112                             System.out.println(currentData.temperature);
113                             gpsAvailable = (Integer.parseInt(splitSentence[2]))>0;
114                             if (gpsAvailable) {
115                                   currentPosition = new
   Position(Double.parseDouble(splitSentence[3]),
116                                                Double.parseDouble(splitSentence[4]));
117                                   gpsAltitude =
   Double.parseDouble(splitSentence[5]);
118                             }
119
120                             packetRecieved = true;
121                             status = Integer.parseInt(splitSentence[6]);
122                             System.out.println(currentData.temperature);
123                             if (status == 7) {
124                                   roverDeployed = DeploymentStatus.DEPLOYED;
125                             } else {
126                                   roverDeployed = DeploymentStatus.LOADED;
127                             }
128
```

```
129                         }
130                   } catch (Exception e) {
131                         e.printStackTrace();
132                   }
133                   ;
134                   if (packetRecieved) {
135                         firePositionUpdate();
136                         fireDataUpdate();
137                         fireStatusUpdate();
138                   }
139             }
140       }
141
142       protected void firePositionUpdate() {
143             for (LandingModuleListener l : listeners) {
144                   l.positionUpdated(currentPosition, this);
145             }
146       }
147
148       protected void fireStatusUpdate() {
149             for (LandingModuleListener l : listeners) {
150                   l.statusUpdated(roverDeployed, connected, this);
151             }
152       }
153
154       protected void fireDataUpdate() {
155             for (LandingModuleListener l : listeners) {
156                   l.dataUpdated(currentData, this);
157             }
158       }
159
160 }
```

6.10 LandingModuleListener.java

```
1  package gui;
2
3  public interface LandingModuleListener {
4        public void positionUpdated(Position newPosition, LandingModule m);
5        public void dataUpdated(TPData currentData, LandingModule m);
6        public void statusUpdated(LandingModule.DeploymentStatus deployed, boolean
   connected, LandingModule m);
7  }
```

6.11 LargeMap.java

```
1  package gui;
2
3  import java.awt.KeyEventDispatcher;
4  import java.awt.KeyboardFocusManager;
5  import java.awt.event.ActionEvent;
6  import java.awt.event.ActionListener;
7  import java.awt.event.KeyEvent;
8  import java.awt.event.KeyListener;
9  import java.awt.event.MouseEvent;
10 import java.awt.event.MouseMotionListener;
11 import java.awt.event.WindowEvent;
12 import java.awt.event.WindowStateListener;
13
```

```
14 import javax.swing.JButton;
15 import javax.swing.JFrame;
16 import javax.swing.JLabel;
17 /*
18  * This class is designed to provide a window showing an enlarged map.
19  */
20 import javax.swing.SpringLayout;
21
22 public class LargeMap extends JFrame implements WindowStateListener {
23       RoverUpdateListener l;
24       private static final long serialVersionUID = -1257523731129487909L;
25       CustomMap map_2;
26       Rover rover;
27       JLabel lblPosition;
28       Position relativeDatumPoint = null;
29       Position lastMousePosition = null;
30       LandingModule module = null;
31       LargeMap(Rover r, LandingModule m, Position centre, int zoomLevel) {
32             super();
33             setTitle("Map");
34             setSize(500, 500);
35             SpringLayout springLayout = new SpringLayout();
36             this.getContentPane().setLayout(springLayout);
37             JButton btnGoto = new JButton("Go To");
38             module = m;
39             btnGoto.addActionListener(new ActionListener() {
40
41                   @Override
42                   public void actionPerformed(ActionEvent arg0) {
43                         GoToDialog dialog = new GoToDialog(null);
44                         GoToDialogResult result = dialog.show(map_2.getPosition()
45                                     .getLat(), map_2.getPosition().getLon());
46                         if (result.success) {
47                               switch (result.selectedOption) {
48                               case GOTO_ROVER:
49                                     map_2.setDisplayPositionByLatLon(
50                                                 rover.currentPosition.getLat(),
51                                                 rover.currentPosition.getLon(),
    map_2.getZoom());
52                                     break;
53                               case GOTO_MODULE:
54                                     map_2.setDisplayPositionByLatLon(
55                                                 module.currentPosition.getLat(),
56                                                 module.currentPosition.getLon(),
    map_2.getZoom());
57                                     break;
58                               case GOTO_LATLONG:
59                                     map_2.setDisplayPositionByLatLon(
60                                                 result.enteredPosition.getLat(),
61                                                 result.enteredPosition.getLon(),
62                                                 map_2.getZoom());
63                                     break;
64                               }
65                         }
66                   }
67             });
68             map_2 = new CustomMap(r);
```

```
69              lblPosition = new JLabel("");
70
71              map_2.setDisplayPositionByLatLon(centre.getLat(), centre.getLon(), zoomLevel);
72              map_2.addMouseMotionListener(new MouseMotionListener() {
73                      @Override
74                      public void mouseMoved(MouseEvent arg0) {
75                              int xCoord = arg0.getX();
76                              int yCoord = arg0.getY();
77                              lastMousePosition = new Position(map_2.getPosition(xCoord,
78                                              yCoord));
79                              if(relativeDatumPoint == null) {
80                                      lblPosition.setText(String.format(
81                                                      "Lat: %.5f, Long: %.5f",
82                                                      lastMousePosition.getLat(),
    lastMousePosition.getLon()));
83                              } else {
84                                      lblPosition.setText(String.format(
85                                                      "Lat: %.5f, Long: %.5f, Rel: %.1fm, %.1f",
86                                                      lastMousePosition.getLat(),
    lastMousePosition.getLon(),
87
    relativeDatumPoint.getDistanceTo(lastMousePosition),
88
    relativeDatumPoint.getBearingTo(lastMousePosition)));
89                              }
90
91                      }
92
93                      @Override
94                      public void mouseDragged(MouseEvent arg0) {
95                              // Not used
96
97                      }
98              });
99
    KeyboardFocusManager.getCurrentKeyboardFocusManager().addKeyEventDispatcher(new
    KeyEventDispatcher() {
100                     @Override
101                     public boolean dispatchKeyEvent(KeyEvent e) {
102                             if(e.getKeyCode() == KeyEvent.VK_S) {
103                                     if(lastMousePosition != null) {
104                                             relativeDatumPoint = lastMousePosition;
105                                             lblPosition.setText(String.format(
106                                                             "Lat: %.5f, Long: %.5f, Rel: %.1fm,
    %.1f",
107                                                             lastMousePosition.getLat(),
    lastMousePosition.getLon(),
108
    relativeDatumPoint.getDistanceTo(lastMousePosition),
109
    relativeDatumPoint.getBearingTo(lastMousePosition)));
110                                     }
111                             }
112                             return true;
113
114                     }
115             });
```

```
116                 springLayout.putConstraint(SpringLayout.NORTH, btnGoto, 10,
117                         SpringLayout.NORTH, this.getContentPane());
118                 springLayout.putConstraint(SpringLayout.WEST, btnGoto, 10,
119                         SpringLayout.WEST, this.getContentPane());
120                 getContentPane().add(btnGoto);
121                 springLayout.putConstraint(SpringLayout.NORTH, lblPosition, 0,
122                         SpringLayout.NORTH, btnGoto);
123                 springLayout.putConstraint(SpringLayout.WEST, lblPosition, 5,
124                         SpringLayout.EAST, btnGoto);
125                 getContentPane().add(lblPosition);
126                 springLayout.putConstraint(SpringLayout.NORTH, map_2, 10,
127                         SpringLayout.SOUTH, btnGoto);
128                 springLayout.putConstraint(SpringLayout.WEST, map_2, 0,
129                         SpringLayout.WEST, this.getContentPane());
130                 springLayout.putConstraint(SpringLayout.EAST, map_2, 0,
131                         SpringLayout.EAST, this.getContentPane());
132                 springLayout.putConstraint(SpringLayout.SOUTH, map_2, 0,
133                         SpringLayout.SOUTH, this.getContentPane());
134                 getContentPane().add(map_2);
135
136                 setVisible(true);
137                 this.rover = r;
138                 module.attachListener(map_2);
139                 rover.attachListener(map_2);
140         }
141
142         @Override
143         public void windowStateChanged(WindowEvent arg0) {
144
145         };
146
147         public void windowClosed(WindowEvent e) {
148                 rover.removeListener(map_2);
149                 module.removeListener(map_2);
150         }
151 }
```

6.12 MapClickHandler.java

```
1  package gui;
2
3  /*
4   * This class handles rover position updating, and is designed to be attached as
5   * a mouse listener to a JMapViewer or CustomMap
6   */
7
8  import java.awt.event.MouseEvent;
9  import java.awt.event.MouseListener;
10
11 import javax.swing.JOptionPane;
12
13 import org.openstreetmap.gui.jmapviewer.JMapViewer;
14
15 public class MapClickHandler implements MouseListener {
16       public MapClickHandler(Rover rover) {
17             super();
18             this.rover = rover;
19       }
```

```
20
21      Rover rover;
22      @Override
23      public void mouseClicked(MouseEvent arg0) {
24            JMapViewer eventOriginator;
25            int xCoord, yCoord;
26            int move;
27            Position realPosition;
28            if(arg0.getButton() == MouseEvent.BUTTON1) {
29                  if(arg0.getComponent() instanceof JMapViewer) { //Basic sanity check
30                        //Obtain the original JMapViewer for position look-up
31                        eventOriginator = (JMapViewer) arg0.getComponent();
32                        //Get raw X and Y coordinates (relative to the JMapViewer object)
33                        xCoord = arg0.getX();
34                        yCoord = arg0.getY();
35                        realPosition = new
   Position(eventOriginator.getPosition(xCoord,yCoord));
36                        //Prompt the user whether they really want to move
37                        move =
   JOptionPane.showConfirmDialog(eventOriginator.getRootPane(),
38                                    "Move to " + realPosition.toString() + "?\n" +
39                                          "Distance: " +
40
   String.format("%.0f",realPosition.getDistanceTo(rover.currentPosition)*1000) +
41                                          "m",
42                                          "Confirm Move",
43                                          JOptionPane.YES_NO_OPTION);
44                        if (move == JOptionPane.YES_OPTION) {
45                              rover.moveToPosition(realPosition);
46                              rover.atTargetPosition = false;
47                        }
48                  }
49            };
50      }
51
52      @Override
53      public void mouseEntered(MouseEvent arg0) {
54            // Not used at the time being
55
56      }
57
58      @Override
59      public void mouseExited(MouseEvent arg0) {
60            // Not used at the time being
61
62      }
63
64      @Override
65      public void mousePressed(MouseEvent arg0) {
66            // Not used at the time being
67
68      }
69
70      @Override
71      public void mouseReleased(MouseEvent arg0) {
72            // Not used at the time being
73
```

```
74      }
75
76  }
```

6.13 NetworkCommand.java

```
1   package gui;
2
3   public class NetworkCommand {
4       private String command, payload;
5       public NetworkCommand() {
6           command = "";
7           payload = "";
8       }
9       public NetworkCommand(String cmd) {
10          command = cmd;
11          payload = "";
12      }
13      public NetworkCommand(String cmd, String data) {
14          command = cmd;
15          payload = data;
16      }
17      public String toString() {
18          String s;
19          s = "HI " + command + " " + payload;
20          return s;
21      }
22  }
```

6.14 NetworkException.java

```
1   package gui;
2
3   public class NetworkException extends Exception {
4
5       private static final long serialVersionUID = 5622726798275639631L;
6       private NetworkStatus errorCode;
7       private String description;
8       public NetworkException() {
9           errorCode = NetworkStatus.UNKNOWN_ERROR;
10          description = "";
11      }
12      public NetworkException(NetworkStatus status) {
13          errorCode = status;
14          description = "";
15      }
16      public NetworkException(NetworkStatus status, String desc) {
17          errorCode = status;
18          description = desc;
19      }
20      public NetworkStatus getErrorCode() {
21          return errorCode;
22      }
23      public String getDescription() {
24          return description;
25      }
26  }
```

6.15 NetworkImageViewer.java

```
1  package gui;
2
3  import java.awt.Color;
4  import java.awt.Graphics;
5  import java.awt.Graphics2D;
6  import java.awt.LayoutManager;
7  import java.awt.geom.AffineTransform;
8  import java.awt.image.BufferedImage;
9  import java.io.IOException;
10 import java.net.MalformedURLException;
11 import java.net.URL;
12 import java.util.Timer;
13 import java.util.TimerTask;
14
15 import javax.imageio.ImageIO;
16 import javax.swing.JPanel;
17
18 public class NetworkImageViewer extends JPanel {
19       private static final long serialVersionUID = -5555045450327984237L;
20       private SettingsStore settings = new SettingsStore();
21       private BufferedImage image = null;
22       private Timer t;
23       //private boolean running = false;
24       public NetworkImageViewer() {
25             super();
26       }
27
28       public NetworkImageViewer(LayoutManager arg0) {
29             super(arg0);
30       }
31
32       public NetworkImageViewer(boolean arg0) {
33             super(arg0);
34       }
35
36       public NetworkImageViewer(LayoutManager arg0, boolean arg1) {
37             super(arg0, arg1);
38       }
39
40       public void startDisplay() {
41             t = new Timer();
42             TimerTask task = new TimerTask() {
43
44                   @Override
45                   public void run() {
46                         updateImage();
47
48                   }
49             };
50             t.scheduleAtFixedRate(task, 0, 2000);
51       }
52
53       public void stopDisplay() {
54             t.cancel();
55       }
```

```
56
57      public void updateImage() {
58            BufferedImage downloadedImage;
59            try {
60                  downloadedImage = ImageIO.read(new URL("http://" +
    settings.get("rover.ip") + "/image.png"));
61                        image = downloadedImage;
62                        //Redraw new image
63                        super.repaint();
64            } catch (MalformedURLException e) {
65                  e.printStackTrace();
66            } catch (IOException e) {
67                  e.printStackTrace();
68            }
69      }
70
71      @Override
72      protected void paintComponent(Graphics g) {
73            super.paintComponent(g);
74            if (image != null) {
75                   AffineTransform at = new AffineTransform();
76           at.translate(getWidth() / 2, getHeight() / 2);
77           at.rotate(Math.PI/2);
78           at.scale(0.5, 0.5);
79           at.translate(-image.getWidth()/2, -image.getHeight()/2);
80           Graphics2D g2d = (Graphics2D) g;
81           g2d.drawImage(image, at, null);
82            }
83            g.setColor(new Color(255,0,0,100));
84            g.fillRect(295, 239, 15, 2);
85            g.fillRect(330, 239, 15, 2);
86            g.fillRect(319, 215, 2, 15);
87            g.fillRect(319, 250, 2, 15);
88      }
89 }
```

6.16 NetworkResponse.java

```
1  package gui;
2
3  public class NetworkResponse {
4       private NetworkStatus status;
5       private String payload;
6       public NetworkResponse(String response) {
7            if(response.length() < 5) {
8                  status = NetworkStatus.COMMUNICATION_ERROR;
9                  payload = "";
10           } else {
11                 if(!response.substring(0, 2).contentEquals("RP")) {
12                       status = NetworkStatus.COMMUNICATION_ERROR;
13                       payload = "";
14                 } else {
15                       if(response.length() < 7) {
16                             payload = "";
17                       } else {
18                             payload = response.substring(6);
19                       }
20
```

```
21                      switch(response.substring(3,5)) {
22                      case "OK":
23                              status = NetworkStatus.OK;
24                              break;
25                      case "CE":
26                              status = NetworkStatus.COMMUNICATION_ERROR;
27                              break;
28                      case "NC":
29                              status = NetworkStatus.UNKNOWN_COMMAND;
30                              break;
31                      case "NP":
32                              status = NetworkStatus.PARAMETER_ERROR;
33                              break;
34                      case "OE":
35                              status = NetworkStatus.ROVER_ERROR;
36                              break;
37                      default:
38                              status = NetworkStatus.UNKNOWN_ERROR;
39                              break;
40                      }
41              }
42      }
43  }
44  public NetworkResponse(NetworkStatus error) {
45          status = error;
46          payload = "";
47  }
48  public String getPayload() {
49          return payload;
50  }
51  public NetworkStatus getStatus() {
52          return status;
53  }
54 }
```

6.17 NetworkRover.java

```
1  package gui;
2
3  import javax.swing.JOptionPane;
4
5  public class NetworkRover extends Rover {
6      private NetworkSender network;
7      private boolean online = false;
8      private boolean busy = false;
9      enum MoveOperation {
10             FORWARDS,
11             BACKWARDS,
12             LEFT_TURN,
13             RIGHT_TURN
14     }
15
16     public NetworkRover() {
17
18     }
19
20     boolean begin(String hostname) {
21             busy = false;
```

```
22              network = new NetworkSender(hostname);
23              online = true;
24              System.err.println(testNetwork().toString());
25              if(testNetwork() == NetworkStatus.OK) {
26                      //Initialise video stream
27                      return true;
28              } else {
29                      disconnect();
30                      return false;
31              }
32      }
33
34      NetworkStatus testNetwork() {
35              if(online) {
36                      return network.sendCommand(new
NetworkCommand("PI")).getStatus();
37              } else {
38                      return NetworkStatus.ROVER_OFFLINE;
39              }
40      }
41
42      void disconnect() {
43              online = false;
44              busy = false;
45              network = null;
46              //Stop video stream
47              //Any other closing down stuff
48      }
49
50      private void waitForFreeNetwork() throws NetworkException {
51              long startWaitTime = System.currentTimeMillis();
52              while(busy) {
53                      if((System.currentTimeMillis() - startWaitTime) > 5000) {
54                              System.err.println("---");
55
56                              throw new NetworkException(NetworkStatus.TIMEOUT);
57                      }
58              }
59      }
60
61      public void moveToPosition(Position p) {
62              if(online) {
63                      try {
64                              waitForFreeNetwork();
65                      } catch (NetworkException e) {
66                              return;
67                      }
68                      busy = true;
69                      targetPosition = p;
70                      atTargetPosition = false;
71                      NetworkCommand cmd = new NetworkCommand("GT",p.getLat() + " "
+ p.getLon());
72                      NetworkStatus status = network.sendCommand(cmd).getStatus();
73                      busy = false;
74                      switch(status) {
75                      case OK:
76                              break;
```

```
77                  case COMMUNICATION_ERROR:
78                      //Try once more
79                      try {
80                          waitForFreeNetwork();
81                      } catch (NetworkException e) {
82                          break;
83                      }
84                      busy = true;
85                      NetworkStatus newStatus =
    network.sendCommand(cmd).getStatus();
86                      busy = false;
87                      if(newStatus == NetworkStatus.OK) break;
88                      //Otherwise drop into error handling
89                  default:
90                      targetPosition = currentPosition;
91                      atTargetPosition = true;
92                      JOptionPane.showMessageDialog(null,"Network error: " +
    status.toString());
93                      break;
94                  }
95                  busy = false;
96              };
97      }
98
99      public Position getPosition() throws NetworkException {
100             if(online) {
101                 waitForFreeNetwork();
102                 busy = true;
103                 NetworkCommand cmd = new NetworkCommand("CL");
104                 NetworkResponse response = network.sendCommand(cmd);
105                 busy = false;
106                 switch(response.getStatus()) {
107                 case OK:
108                     break;
109                 case COMMUNICATION_ERROR:
110                   //Try once more
111                     waitForFreeNetwork();
112                     busy = true;
113                     NetworkStatus newStatus =
    network.sendCommand(cmd).getStatus();
114                     busy = false;
115                     if(newStatus == NetworkStatus.OK) break;
116                   //Otherwise drop into error handling
117                 default:
118                     targetPosition = currentPosition;
119                     atTargetPosition = true;
120                     throw new NetworkException(response.getStatus());
121                 }
122                 String splitStr[] = response.getPayload().split("\\s+");
123                 if(splitStr.length < 2) throw new
    NetworkException(NetworkStatus.RESPONSE_INVALID,
124                                     "Position string returned: " + response.getPayload());
125                 try {
126                     double lat = Double.parseDouble(splitStr[0]);
127                     double lon = Double.parseDouble(splitStr[1]);
128                     return new Position(lat,lon);
129                 } catch(Exception e) {
```

```
130                         throw new
    NetworkException(NetworkStatus.RESPONSE_INVALID,"Position string returned: " +
    response.getPayload());
131                     }
132                 } else {
133
134                     throw new NetworkException(NetworkStatus.ROVER_OFFLINE);
135                 }
136             }
137
138             public void update() {
139                 if(online) {
140                   //Test connectivity?
141                     try {
142                         currentPosition = getPosition();
143                         firePositionUpdate();
144                     } catch (NetworkException e) {
145                         throwError("Couldn't update position - error: " +
    e.getErrorCode().toString());
146                     }
147
148                     try {
149                         currentData = getTP();
150                         fireDataUpdate();
151                     } catch (NetworkException e) {
152                         throwError("Couldn't update TP - error: " +
    e.getErrorCode().toString());
153                     }
154                 }
155             }
156
157             public void moveManually(MoveOperation how) {
158                 if(online) {
159                     String opcode = "";
160                     switch(how) {
161                     case FORWARDS:
162                         opcode = "F";
163                         break;
164                     case BACKWARDS:
165                         opcode = "B";
166                         break;
167                     case LEFT_TURN:
168                         opcode = "L";
169                         break;
170                     case RIGHT_TURN:
171                         opcode = "R";
172                         break;
173                     }
174                     try {
175                         waitForFreeNetwork();
176                     } catch(NetworkException e) {
177                         throwError("Couldn't move - error: timeout");
178                         return;
179                     }
180                     busy = true;
181                     NetworkCommand cmd = new NetworkCommand("MV",opcode);
182                     NetworkStatus status = network.sendCommand(cmd).getStatus();
```

```
183                     busy = false;
184                     switch(status) {
185                     case OK:
186                             break;
187                     case COMMUNICATION_ERROR:
188                       //Try once more
189                             try {
190                                     waitForFreeNetwork();
191                             } catch(NetworkException e) {
192                                     throwError("Couldn't move - error: timeout");
193                                     return;
194                             }
195                             busy = true;
196                             NetworkStatus newStatus =
    network.sendCommand(cmd).getStatus();
197                             busy = false;
198                             if(newStatus == NetworkStatus.OK) break;
199                       //Otherwise drop into error handling
200                     default:
201                             throwError("Couldn't move - error: " + status.toString());
202                             return;
203                     }
204                     targetPosition = currentPosition;
205                     atTargetPosition = true;
206             };
207     }
208
209     public void stopRover() {
210             if(online) {
211                     try {
212                             waitForFreeNetwork();
213                     } catch(NetworkException e) {
214                             throwError("Couldn't stop - error: timeout");
215                             return;
216                     }
217                     busy = true;
218                     NetworkCommand cmd = new NetworkCommand("ST");
219                     NetworkStatus status = network.sendCommand(cmd).getStatus();
220                     busy = false;
221                     switch(status) {
222                     case OK:
223                             break;
224                     case COMMUNICATION_ERROR:
225                       //Try once more
226                             try {
227                                     waitForFreeNetwork();
228                             } catch(NetworkException e) {
229                                     throwError("Couldn't stop - error: timeout");
230                                     return;
231                             }
232                             busy = true;
233                             NetworkStatus newStatus =
    network.sendCommand(cmd).getStatus();
234                             busy = false;
235                             if(newStatus == NetworkStatus.OK) break;
236                       //Otherwise drop into error handling
237                     default:
```

```
238                         throwError("Couldn't stop - error: " + status.toString());
239                         return;
240                   }
241                   targetPosition = currentPosition;
242                   atTargetPosition = true;
243             };
244       }
245
246       public void formatEEPROM() {
247             if(online) {
248                   try {
249                         waitForFreeNetwork();
250                   } catch(NetworkException e) {
251                         throwError("Couldn't format EEPROM - error: timeout");
252                         return;
253                   }
254                   busy = true;
255                   NetworkCommand cmd = new NetworkCommand("FT");
256                   NetworkStatus status = network.sendCommand(cmd).getStatus();
257                   busy = false;
258                   switch(status) {
259                   case OK:
260                         break;
261                   case COMMUNICATION_ERROR:
262                     //Try once more
263                         try {
264                               waitForFreeNetwork();
265                         } catch(NetworkException e) {
266                               throwError("Couldn't format EEPROM - error:
    timeout");
267                               return;
268                         }
269                         busy = true;
270                         NetworkStatus newStatus =
    network.sendCommand(cmd).getStatus();
271                         busy = false;
272                         if(newStatus == NetworkStatus.OK) break;
273                     //Otherwise drop into error handling
274                   default:
275                         throwError("Couldn't format EEPROM - error: " +
    status.toString());
276                         return;
277                   }
278             };
279       }
280
281       public TPData getTP() throws NetworkException {
282             if(online) {
283                   waitForFreeNetwork();
284                   busy = true;
285                   NetworkCommand cmd = new NetworkCommand("TP");
286                   NetworkResponse response = network.sendCommand(cmd);
287                   busy = false;
288                   switch(response.getStatus()) {
289                   case OK:
290                         break;
291                   case COMMUNICATION_ERROR:
```

```
292                         //Try once more
293                             waitForFreeNetwork();
294                             busy = true;
295                             NetworkStatus newStatus =
    network.sendCommand(cmd).getStatus();
296                             busy = false;
297                             if(newStatus == NetworkStatus.OK) break;
298                         //Otherwise drop into error handling
299                     default:
300                             targetPosition = currentPosition;
301                             atTargetPosition = true;
302                             throw new NetworkException(response.getStatus());
303                     }
304                     String splitStr[] = response.getPayload().split("\\s+");
305                     if(splitStr.length < 2) throw new
    NetworkException(NetworkStatus.RESPONSE_INVALID,
306                                     "TP string returned: " + response.getPayload());
307                     try {
308                             double t = Double.parseDouble(splitStr[0]);
309                             double p = Double.parseDouble(splitStr[1]);
310                             return new TPData(t,p);
311                     } catch(Exception e) {
312                             throw new
    NetworkException(NetworkStatus.RESPONSE_INVALID,"TP string returned: " +
    response.getPayload());
313                     }
314             } else {
315                     throw new NetworkException(NetworkStatus.ROVER_OFFLINE);
316             }
317         }
318     }
```

6.18 NetworkSender.java

```
1   package gui;
2
3   import java.io.BufferedReader;
4   import java.io.InputStreamReader;
5   import java.io.PrintWriter;
6   import java.net.Socket;
7   import java.net.SocketTimeoutException;
8
9   public class NetworkSender {
10      private Socket roverSocket;
11      private String roverHostname;
12
13      private PrintWriter out;
14      private BufferedReader in;
15      public NetworkSender(String hostname) {
16              roverHostname = hostname;
17      }
18      private boolean connect() {
19              try {
20                      disconnect();
21                      roverSocket = new Socket(roverHostname,9001);
22                      roverSocket.setSoTimeout(2500);
23                      out = new PrintWriter(roverSocket.getOutputStream(),true);
```

```
24                  in = new BufferedReader(new
    InputStreamReader(roverSocket.getInputStream()));
25
26                  return true;
27            } catch(Exception e) {
28                  return false;
29            }
30      }
31
32      private void disconnect() {
33            try {
34                  roverSocket.close();
35            } catch (Exception e) {
36
37            }
38      }
39      public NetworkResponse sendCommand(NetworkCommand command) {
40            if(connect()) {
41                  try {
42                        out.println(command.toString());
43                        String responseText = in.readLine();
44                        return new NetworkResponse(responseText);
45                  } catch (SocketTimeoutException e) {
46                        return new NetworkResponse(NetworkStatus.TIMEOUT);
47                  } catch (Exception e) {
48                        return new
    NetworkResponse(NetworkStatus.COMMUNICATION_ERROR);
49                  }
50            } else {
51                  return new NetworkResponse(NetworkStatus.ROVER_OFFLINE);
52            }
53      }
54
55 }
```

6.19 NetworkStatus.java

```
1  package gui;
2
3  public enum NetworkStatus {
4        OK,
5        COMMUNICATION_ERROR,
6        UNKNOWN_COMMAND,
7        PARAMETER_ERROR,
8        ROVER_ERROR,
9        ROVER_OFFLINE,
10       TIMEOUT,
11       UNKNOWN_ERROR,
12       RESPONSE_INVALID
13 }
```

6.20 Position.java

```
1  package gui;
2
3  import org.openstreetmap.gui.jmapviewer.Coordinate;
4
5  /*
6   * Provides a class for storing a latitude/longitude pair, as well
```

```
7   * as distance calculations to another position
8   */
9
10  public class Position {
11      private double latitude, longitude;
12      public Position(double latitude, double longitude) {
13          super();
14          this.latitude = latitude;
15          this.longitude = longitude;
16      }
17      //A position class can be initialised with a JMapViewer Coordinate.
18      public Position(Coordinate c) {
19          super();
20          this.latitude = c.getLat();
21          this.longitude = c.getLon();
22      }
23      public Position(Position p) {
24          super();
25          this.latitude = p.getLat();
26          this.longitude = p.getLon();
27      }
28      public void set(double lat, double lon ) {
29          latitude = lat;
30          longitude = lon;
31      }
32      public double getLat() {
33          return latitude;
34      }
35      public double getLon() {
36          return longitude;
37      }
38
39      public void addOffset(double deltaLat, double deltaLon) {
40          latitude += deltaLat;
41          longitude += deltaLon;
42      }
43
44      //Vincenty formula for calculating distance
45      //Converted from a Javascript implementation
46      public double getDistanceTo(Position p) {
47          double a = 6378137, b = 6356752.314245,  f = 1/298.257223563;  //
    WGS-84 ellipsoid params
48          double L = Math.toRadians((p.longitude-longitude));
49          double U1 = Math.atan((1-f) * Math.tan(Math.toRadians(latitude)));
50          double U2 = Math.atan((1-f) * Math.tan(Math.toRadians(p.latitude)));
51          double sinU1 = Math.sin(U1), cosU1 = Math.cos(U1);
52          double sinU2 = Math.sin(U2), cosU2 = Math.cos(U2);
53          double cosSqAlpha, sinSigma, cos2SigmaM, cosSigma, sigma;
54          double lambda = L, lambdaP, iterLimit = 100;
55          do {
56            double sinLambda = Math.sin(lambda), cosLambda = Math.cos(lambda);
57            sinSigma = Math.sqrt((cosU2*sinLambda) * (cosU2*sinLambda) +
58              (cosU1*sinU2-sinU1*cosU2*cosLambda) * (cosU1*sinU2-
    sinU1*cosU2*cosLambda));
59            if (sinSigma==0) return 0;  // co-incident points
60            cosSigma = sinU1*sinU2 + cosU1*cosU2*cosLambda;
61            sigma = Math.atan2(sinSigma, cosSigma);
```

```
62              double sinAlpha = cosU1 * cosU2 * sinLambda / sinSigma;
63              cosSqAlpha = 1 - sinAlpha*sinAlpha;
64              cos2SigmaM = cosSigma - 2*sinU1*sinU2/cosSqAlpha;
65              if ((cos2SigmaM) == Double.NaN) cos2SigmaM = 0;  // equatorial line:
    cosSqAlpha=0
66              double C = f/16*cosSqAlpha*(4+f*(4-3*cosSqAlpha));
67              lambdaP = lambda;
68              lambda = L + (1-C) * f * sinAlpha *
69                (sigma + C*sinSigma*(cos2SigmaM+C*cosSigma*(-
    1+2*cos2SigmaM*cos2SigmaM)));
70            } while (Math.abs(lambda-lambdaP) > 1e-12 && --iterLimit>0);
71
72            if (iterLimit==0) return -1;  // formula failed to converge
73
74            double uSq = cosSqAlpha * (a*a - b*b) / (b*b);
75            double A = 1 + uSq/16384*(4096+uSq*(-768+uSq*(320-175*uSq)));
76            double B = uSq/1024 * (256+uSq*(-128+uSq*(74-47*uSq)));
77            double deltaSigma = B*sinSigma*(cos2SigmaM+B/4*(cosSigma*(-
    1+2*cos2SigmaM*cos2SigmaM)-
78              B/6*cos2SigmaM*(-3+4*sinSigma*sinSigma)*(-
    3+4*cos2SigmaM*cos2SigmaM)));
79            double s = b*A*(sigma-deltaSigma);
80            return s;
81        }
82
83        public double getBearingTo(Position p) {
84            double  f = 1/298.257223563;  // WGS-84 ellipsoid params
85            double L = Math.toRadians((p.longitude-longitude));
86            double U1 = Math.atan((1-f) * Math.tan(Math.toRadians(latitude)));
87            double U2 = Math.atan((1-f) * Math.tan(Math.toRadians(p.latitude)));
88            double sinU1 = Math.sin(U1), cosU1 = Math.cos(U1);
89            double sinU2 = Math.sin(U2), cosU2 = Math.cos(U2);
90            double cosSqAlpha, sinSigma, cos2SigmaM, cosSigma, sigma, sinLambda,
    cosLambda;
91            double lambda = L, lambdaP, iterLimit = 100;
92            do {
93              sinLambda = Math.sin(lambda);
94              cosLambda = Math.cos(lambda);
95              sinSigma = Math.sqrt((cosU2*sinLambda) * (cosU2*sinLambda) +
96                (cosU1*sinU2-sinU1*cosU2*cosLambda) * (cosU1*sinU2-
    sinU1*cosU2*cosLambda));
97              if (sinSigma==0) return 0;  // co-incident points
98              cosSigma = sinU1*sinU2 + cosU1*cosU2*cosLambda;
99              sigma = Math.atan2(sinSigma, cosSigma);
100                 double sinAlpha = cosU1 * cosU2 * sinLambda / sinSigma;
101                 cosSqAlpha = 1 - sinAlpha*sinAlpha;
102                 cos2SigmaM = cosSigma - 2*sinU1*sinU2/cosSqAlpha;
103             if ((cos2SigmaM) == Double.NaN) cos2SigmaM = 0;  // equatorial line:
    cosSqAlpha=0
104                 double C = f/16*cosSqAlpha*(4+f*(4-3*cosSqAlpha));
105                 lambdaP = lambda;
106                 lambda = L + (1-C) * f * sinAlpha *
107                   (sigma + C*sinSigma*(cos2SigmaM+C*cosSigma*(-
    1+2*cos2SigmaM*cos2SigmaM)));
108               } while (Math.abs(lambda-lambdaP) > 1e-12 && --iterLimit>0);
109
110             if (iterLimit==0) return -1;  // formula failed to converge
```

```
111
112                 double fwdAz = Math.atan2(cosU2*sinLambda,  cosU1*sinU2-
    sinU1*cosU2*cosLambda);
113                 return Math.toDegrees(fwdAz);
114         }
115
116         public String toString() {
117                 String s;
118                 s = String.format("%.5f, %.5f",latitude,longitude);
119                 return s;
120         };
121     }
```

6.21 Rover.java

```
1   package gui;
2
3   import gui.NetworkRover.MoveOperation;
4
5   import java.util.Vector;
6
7   //Provides a base class that handles the calling RoverUpdateListeners
8
9   public class Rover {
10      Vector<RoverUpdateListener> listeners = new Vector<RoverUpdateListener>();
11      public Position currentPosition, targetPosition;
12      public boolean atTargetPosition;
13      protected TPData currentData;
14
15      public Rover() {
16              currentPosition = new Position(0,0);
17              targetPosition = new Position(0,0);
18              atTargetPosition = true;
19      }
20      public void attachListener(RoverUpdateListener l) {
21              listeners.add(l);
22      };
23      public void removeListener(RoverUpdateListener l) {
24              listeners.remove(l);
25      };
26
27
28      protected void firePositionUpdate() {
29              for(RoverUpdateListener l : listeners) {
30                      l.positionUpdated(currentPosition, targetPosition, atTargetPosition, this);
31              }
32      }
33
34      protected void fireDataUpdate() {
35              for(RoverUpdateListener l : listeners) {
36                      l.dataUpdated(currentData, this);
37              }
38      }
39
40      protected void forwardMessage(String message) {
41              for(RoverUpdateListener l : listeners) {
42                      l.messageRecieved(message, this);
43              }
```

```
44      };
45
46      protected void throwError(String message) {
47          for(RoverUpdateListener l : listeners) {
48              l.errorThrown(message, this);
49          }
50      }
51
52      public void moveToPosition(Position p) {
53
54      }
55
56      public void update() {
57
58      }
59
60      public void moveManually(MoveOperation how) {
61
62      }
63  }
```

6.22 RoverUpdateListener.java

```
1  package gui;
2
3  public interface RoverUpdateListener {
4      public void positionUpdated(Position newPosition, Position targetPosition, boolean
   atTargetPosition, Rover r);
5      public void dataUpdated(TPData data, Rover r);
6      public void messageRecieved(String message, Rover r);
7      public void errorThrown(String message, Rover r);
8
9  }
```

6.23 SettingsStore.java

```
1  package gui;
2
3  import java.io.BufferedReader;
4  import java.io.FileInputStream;
5  import java.io.FileOutputStream;
6  import java.io.InputStream;
7  import java.io.InputStreamReader;
8  import java.io.PrintWriter;
9  import java.util.HashMap;
10 import java.util.Map;
11 import java.util.Map.Entry;
12
13 public class SettingsStore {
14     private static final Map<String, String> settings = new HashMap<String,
   String>();
15     private static boolean initialised = false;
16     public SettingsStore() {
17         if (!initialised) {
18             initialised = true;
19             InputStream inputStream;
20             BufferedReader br;
21             String line;
22             try {
```

```
23                      inputStream = new FileInputStream("settings.txt");
24                      br = new BufferedReader(new InputStreamReader(inputStream));
25                      while ((line = br.readLine()) != null) {
26                              line = line.trim();
27                              if(line.charAt(0) == '#') continue;
28                              String[] splitLine = line.split(" ",2);
29                              if(splitLine.length > 1) {
30                                      settings.put(splitLine[0],splitLine[1]);
31                              };
32                      }
33                      br.close();
34                      br = null;
35                      inputStream = null;
36              } catch(Exception e) {
37
38              }
39              boolean changed = false;
40              if(!settings.containsKey("rover.ip")) {
41                      settings.put("rover.ip", "0.0.0.0");
42                      changed = true;
43              }
44              if(!settings.containsKey("serial.port")) {
45                      settings.put("serial.port", "COM0");
46                      changed = true;
47              }
48              if(!settings.containsKey("serial.baud")) {
49                      settings.put("serial.baud", "57600");
50                      changed = true;
51              }
52
53              if(!settings.containsKey("home.lat")) {
54                      settings.put("home.lat", "51.487556");
55                      changed = true;
56              }
57
58              if(!settings.containsKey("home.long")) {
59                      settings.put("home.long", "-0.2381855");
60                      changed = true;
61              }
62
63              if(!settings.containsKey("home.alt")) {
64                      settings.put("home.alt", "0");
65                      changed = true;
66              }
67              if(changed) {
68                      save();
69              }
70      }
71
72  }
73  public void save() {
74          FileOutputStream outputStream;
75          PrintWriter fileWriter;
76          try {
77                  outputStream = new FileOutputStream("settings.txt");
78                  fileWriter = new PrintWriter(outputStream);
79                  for(Entry<String, String> entry : settings.entrySet()) {
```

```
80                      fileWriter.println(entry.getKey() + " " +entry.getValue());
81                  }
82                  fileWriter.close();
83                  outputStream.close();
84                  outputStream = null;
85                  fileWriter = null;
86              } catch(Exception e) {
87
88              }
89      }
90      public void set(String key, String value) {
91              settings.put(key, value);
92      }
93      public String get(String key) {
94              if(settings.containsKey(key)) {
95                      return settings.get(key);
96              } else {
97                      return null;
98              }
99      }
100         public double getDouble(String key) {
101                 try {
102                         return Double.parseDouble(settings.get(key));
103                 } catch(Exception e) {
104                         return 0;
105                 }
106         }
107         public int getInt(String key) {
108                 try {
109                         return Integer.parseInt(settings.get(key));
110                 } catch(Exception e) {
111                         return 0;
112                 }
113         }
114 }
```

6.24 SettingsWindow.java

```
1  package gui;
2
3  import java.awt.event.ActionEvent;
4  import java.awt.event.ActionListener;
5  import java.awt.event.WindowEvent;
6  import java.util.Enumeration;
7
8  import javax.swing.JButton;
9  import javax.swing.JComboBox;
10 import javax.swing.JFrame;
11 import javax.swing.JLabel;
12 import javax.swing.JSeparator;
13 import javax.swing.JTextField;
14 import javax.swing.SpringLayout;
15
16 import gnu.io.CommPortIdentifier;
17
18 public class SettingsWindow extends JFrame {
19     private static final long serialVersionUID = 8536169689704756762L;
20     private SettingsStore settings;
```

```
21      SettingsWindow() {
22          super();
23          settings  =new SettingsStore();
24          setTitle("Nova - Settings");
25          setSize(500, 250);
26          SpringLayout springLayout = new SpringLayout();
27          this.getContentPane().setLayout(springLayout);
28          JLabel lblRoverIP = new JLabel("Rover IP address: ");
29          springLayout.putConstraint(SpringLayout.NORTH, lblRoverIP, 10,
    SpringLayout.NORTH, this);
30          springLayout.putConstraint(SpringLayout.WEST, lblRoverIP, 5,
    SpringLayout.WEST, this);
31          this.add(lblRoverIP);
32          final JTextField txtRoverIP = new JTextField(settings.get("rover.ip"));
33          springLayout.putConstraint(SpringLayout.NORTH, txtRoverIP, 10,
    SpringLayout.NORTH, this);
34          springLayout.putConstraint(SpringLayout.WEST, txtRoverIP, 0,
    SpringLayout.EAST, lblRoverIP);
35          springLayout.putConstraint(SpringLayout.EAST, txtRoverIP, 200,
    SpringLayout.EAST, lblRoverIP);
36          this.add(txtRoverIP);
37
38          JSeparator separator1 = new JSeparator();
39          springLayout.putConstraint(SpringLayout.NORTH, separator1, 10,
    SpringLayout.SOUTH, txtRoverIP);
40          springLayout.putConstraint(SpringLayout.WEST, separator1, 0,
    SpringLayout.WEST, this.getContentPane());
41          springLayout.putConstraint(SpringLayout.EAST, separator1, 0,
    SpringLayout.EAST, this.getContentPane());
42          springLayout.putConstraint(SpringLayout.SOUTH, separator1, 12,
    SpringLayout.SOUTH, txtRoverIP);
43
44          this.add(separator1);
45
46          JLabel lblSerialPort = new JLabel("Arduino Serial Port: ");
47          springLayout.putConstraint(SpringLayout.NORTH, lblSerialPort, 10,
    SpringLayout.NORTH, separator1);
48          springLayout.putConstraint(SpringLayout.WEST, lblSerialPort, 5,
    SpringLayout.WEST, this);
49          this.add(lblSerialPort);
50
51          final JComboBox<String> serialPortList = new JComboBox<String>();
52          @SuppressWarnings("rawtypes")
53          Enumeration portList = CommPortIdentifier.getPortIdentifiers();
54          boolean foundPort = false;
55        while (portList.hasMoreElements()) {
56          CommPortIdentifier portId = (CommPortIdentifier) portList.nextElement();
57          if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
58            serialPortList.addItem(portId.getName());
59            if(portId.getName().equals(settings.get("serial.port"))) {
60              foundPort = true;
61              serialPortList.setSelectedItem(portId.getName());
62            }
63          }
64        }
65
66        if(!foundPort) {
```

```
67              serialPortList.addItem(settings.get("serial.port") + " (unavailable)");
68              serialPortList.setSelectedIndex(serialPortList.getItemCount()-1);
69          }
70
71              springLayout.putConstraint(SpringLayout.NORTH, serialPortList, 10,
    SpringLayout.NORTH, separator1);
72              springLayout.putConstraint(SpringLayout.WEST, serialPortList, 10,
    SpringLayout.EAST, lblSerialPort);
73          this.add(serialPortList);
74
75          JButton btnRescan = new JButton("Rescan");
76          btnRescan.addActionListener(new ActionListener() {
77
78                  @Override
79                  public void actionPerformed(ActionEvent arg0) {
80                      serialPortList.removeAllItems();
81                      @SuppressWarnings("rawtypes")
82                      Enumeration portList = CommPortIdentifier.getPortIdentifiers();
83
84                    while (portList.hasMoreElements()) {
85                      CommPortIdentifier portId = (CommPortIdentifier)
    portList.nextElement();
86                      if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL) {
87                          serialPortList.addItem(portId.getName());
88                      }
89                    }
90                  }
91              });
92
93              springLayout.putConstraint(SpringLayout.NORTH, btnRescan, 10,
    SpringLayout.NORTH, separator1);
94              springLayout.putConstraint(SpringLayout.WEST, btnRescan, 10,
    SpringLayout.EAST, serialPortList);
95          this.add(btnRescan);
96
97          JLabel lblBaudRate = new JLabel("Baud rate: ");
98              springLayout.putConstraint(SpringLayout.NORTH, lblBaudRate, 10,
    SpringLayout.SOUTH, serialPortList);
99              springLayout.putConstraint(SpringLayout.WEST, lblBaudRate, 5,
    SpringLayout.WEST, this);
100             this.add(lblBaudRate);
101
102             final JTextField txtBaudRate = new
    JTextField(settings.get("serial.baud"));
103             springLayout.putConstraint(SpringLayout.NORTH, txtBaudRate, 10,
    SpringLayout.SOUTH, serialPortList);
104             springLayout.putConstraint(SpringLayout.WEST, txtBaudRate, 5,
    SpringLayout.EAST, lblBaudRate);
105             springLayout.putConstraint(SpringLayout.EAST, txtBaudRate, 95,
    SpringLayout.EAST, lblBaudRate);
106
107             this.add(txtBaudRate);
108
109             JSeparator separator2 = new JSeparator();
110             springLayout.putConstraint(SpringLayout.NORTH, separator2, 10,
    SpringLayout.SOUTH, txtBaudRate);
```

```
111             springLayout.putConstraint(SpringLayout.WEST, separator2, 0,
    SpringLayout.WEST, this.getContentPane());
112             springLayout.putConstraint(SpringLayout.EAST, separator2, 0,
    SpringLayout.EAST, this.getContentPane());
113             springLayout.putConstraint(SpringLayout.SOUTH, separator2, 12,
    SpringLayout.SOUTH, txtBaudRate);
114
115             this.add(separator2);
116
117             JLabel lblHomeLocation = new JLabel("Home location");
118             springLayout.putConstraint(SpringLayout.NORTH, lblHomeLocation, 10,
    SpringLayout.NORTH, separator2);
119             springLayout.putConstraint(SpringLayout.WEST, lblHomeLocation, 5,
    SpringLayout.WEST, this);
120             this.add(lblHomeLocation);
121
122             JLabel lblHomeLat = new JLabel("Lat: ");
123             springLayout.putConstraint(SpringLayout.NORTH, lblHomeLat, 5,
    SpringLayout.SOUTH, lblHomeLocation);
124             springLayout.putConstraint(SpringLayout.WEST, lblHomeLat, 5,
    SpringLayout.WEST, this);
125             this.add(lblHomeLat);
126
127             final JTextField txtHomeLat = new JTextField(settings.get("home.lat"));
128             springLayout.putConstraint(SpringLayout.NORTH, txtHomeLat, 5,
    SpringLayout.SOUTH, lblHomeLocation);
129             springLayout.putConstraint(SpringLayout.WEST, txtHomeLat, 5,
    SpringLayout.EAST, lblHomeLat);
130             springLayout.putConstraint(SpringLayout.EAST, txtHomeLat, 95,
    SpringLayout.EAST, lblHomeLat);
131
132             this.add(txtHomeLat);
133
134             JLabel lblHomeLong = new JLabel("Long: ");
135             springLayout.putConstraint(SpringLayout.NORTH, lblHomeLong, 5,
    SpringLayout.SOUTH, lblHomeLocation);
136             springLayout.putConstraint(SpringLayout.WEST, lblHomeLong, 10,
    SpringLayout.EAST, txtHomeLat);
137             this.add(lblHomeLong);
138
139             final JTextField txtHomeLong = new
    JTextField(settings.get("home.long"));
140             springLayout.putConstraint(SpringLayout.NORTH, txtHomeLong, 5,
    SpringLayout.SOUTH, lblHomeLocation);
141             springLayout.putConstraint(SpringLayout.WEST, txtHomeLong, 5,
    SpringLayout.EAST, lblHomeLong);
142             springLayout.putConstraint(SpringLayout.EAST, txtHomeLong, 95,
    SpringLayout.EAST, lblHomeLong);
143
144             this.add(txtHomeLong);
145
146             JLabel lblHomeAlt = new JLabel("Alt: ");
147             springLayout.putConstraint(SpringLayout.NORTH, lblHomeAlt, 5,
    SpringLayout.SOUTH, lblHomeLocation);
148             springLayout.putConstraint(SpringLayout.WEST, lblHomeAlt, 10,
    SpringLayout.EAST, txtHomeLong);
149             this.add(lblHomeAlt);
```

```
150
151             final JTextField txtHomeAlt = new JTextField(settings.get("home.alt"));
152             springLayout.putConstraint(SpringLayout.NORTH, txtHomeAlt, 5,
    SpringLayout.SOUTH, lblHomeLocation);
153             springLayout.putConstraint(SpringLayout.WEST, txtHomeAlt, 5,
    SpringLayout.EAST, lblHomeAlt);
154             springLayout.putConstraint(SpringLayout.EAST, txtHomeAlt, 95,
    SpringLayout.EAST, lblHomeAlt);
155
156             this.add(txtHomeAlt);
157
158             JButton btnSaveExit = new JButton("Save & Exit");
159             springLayout.putConstraint(SpringLayout.SOUTH, btnSaveExit, -5,
    SpringLayout.SOUTH, this.getContentPane());
160             springLayout.putConstraint(SpringLayout.WEST, btnSaveExit, 5,
    SpringLayout.WEST, this);
161             setVisible(true);
162             btnSaveExit.addActionListener(new ActionListener() {
163                 @Override
164                 public void actionPerformed(ActionEvent arg0) {
165                     settings.set("rover.ip",txtRoverIP.getText().trim());
166                     String port = serialPortList.getSelectedItem().toString();
167                     if(port.endsWith("(unavailable)")) {
168                         port = port.replaceFirst("\\(unavailable\\)", "").trim();
169                     }
170                     settings.set("serial.port",port);
171                     settings.set("serial.baud",txtBaudRate.getText().trim());
172                     settings.set("home.lat",txtHomeLat.getText().trim());
173                     settings.set("home.long",txtHomeLong.getText().trim());
174                     settings.set("home.alt",txtHomeAlt.getText().trim());
175                     settings.save();
176                     dispatchEvent(new WindowEvent(SettingsWindow.this,
    WindowEvent.WINDOW_CLOSING));
177
178                 }
179             });
180             this.add(btnSaveExit);
181
182             JButton btnCancel = new JButton("Cancel");
183             springLayout.putConstraint(SpringLayout.SOUTH, btnCancel, -5,
    SpringLayout.SOUTH, this.getContentPane());
184             springLayout.putConstraint(SpringLayout.WEST, btnCancel, 10,
    SpringLayout.EAST, btnSaveExit);
185             btnCancel.addActionListener(new ActionListener() {
186
187                 @Override
188                 public void actionPerformed(ActionEvent arg0) {
189                     dispatchEvent(new WindowEvent(SettingsWindow.this,
    WindowEvent.WINDOW_CLOSING));
190                 }
191             });
192             setVisible(true);
193
194             this.add(btnCancel);
195
196
197         }
```

```
198     }
```

6.25 TPData.java

```
1   package gui;
2
3   public class TPData {
4         public double temperature, pressure;
5         public TPData() {
6
7         }
8         public TPData(double t, double p) {
9               temperature = t;
10              pressure = p;
11        }
12        public String toString() {
13              return String.format("%.1fC, %.0fPa",temperature,pressure);
14        }
15  }
```

7 Base Station PC – C# Serial Port Wrapper

Program.cs

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6   using System.IO.Ports;
7   namespace SerialWrapper
8   {
9      class Program
10     {
11        static void Main(string[] args)
12        {
13           String portName = Console.ReadLine();
14           SerialPort p = new SerialPort(portName);
15           p.BaudRate = 57600;
16           p.Parity = Parity.None;
17           p.Open();
18           while (true)
19           {
20              String d = Console.ReadLine();
21              if (d.Equals("Q")) break;
22              p.Write(d + "#");
23              if (d.Equals("R"))
24              {
25                 String s = "";
26                 while (!s.Contains("-----END-----"))
27                 {
28                    s = p.ReadLine();
29                    Console.WriteLine(s);
30                 }
31              }
32           }
33        }
34     }
```

```
35 }
```

8 Base Station PC – C# Loader and Updater

8.1 Form1.cs

```
1  using System;
2  using System.Collections.Generic;
3  using System.ComponentModel;
4  using System.Data;
5  using System.Drawing;
6  using System.Linq;
7  using System.Text;
8  using System.Threading.Tasks;
9  using System.Windows.Forms;
10 using System.Diagnostics;
11 using System.IO;
12 using System.IO.Compression;
13 using System.Net;
14 using System.Net.NetworkInformation;
15 using System.Runtime.InteropServices;
16 using IWshRuntimeLibrary;
17 namespace Nova
18 {
19
20    public struct Resource
21    {
22       public Resource(string r_name, bool r_extract, bool r_overwrite)
23       {
24          name = r_name;
25          extract = r_extract;
26          overwrite = r_overwrite;
27       }
28       public string name;
29       public bool extract;
30       public bool overwrite;
31    }
32    enum JavaInstallStatus
33    {
34       JAVA_32,
35       JAVA_64,
36       NONE
37    };
38    public partial class Form1 : Form
39    {
40       const string baseUrl = "http://davidstech.net/builds/";
41       private Resource[] resources;
42       JavaInstallStatus javaInstalled;
43       bool firstRun;
44       bool online;
45       [DllImport("kernel32.dll", SetLastError = true)]
46       public static extern bool Wow64DisableWow64FsRedirection(ref IntPtr ptr);
47       [DllImport("USER32.DLL")]
48       public static extern bool SetForegroundWindow(IntPtr hWnd);
49       System.Timers.Timer t;
50
51       public Form1()
```

```
52      {
53          IntPtr ptr = new IntPtr();
54          Wow64DisableWow64FsRedirection(ref ptr);
55          InitializeComponent();
56      }
57      JavaInstallStatus getJavaVersion()
58      {
59          Process myProcess = null;
60
61          try
62          {
63              var processInfo = new ProcessStartInfo("java.exe", "-d64 -version")
64              {
65                  CreateNoWindow = true,
66                  UseShellExecute = false
67              };
68              myProcess = Process.Start(processInfo);
69              do { } while (!myProcess.WaitForExit(50));
70              if (myProcess.ExitCode != 1)
71              {
72                  return JavaInstallStatus.JAVA_64;
73              }
74              processInfo = new ProcessStartInfo("java.exe", "-version")
75              {
76                  CreateNoWindow = true,
77                  UseShellExecute = false
78              };
79              myProcess = Process.Start(processInfo);
80              do { } while (!myProcess.WaitForExit(50));
81              if (myProcess.ExitCode != 1)
82              {
83                  return JavaInstallStatus.JAVA_32;
84              }
85              return JavaInstallStatus.NONE;
86          }
87          catch(Exception exception)
88          {
89              MessageBox.Show(exception.ToString());
90              return JavaInstallStatus.NONE;
91          }
92
93
94      }
95
96      private bool needsInstall(Resource r)
97      {
98          if (!System.IO.File.Exists(r.name))
99          {
100                 return true;
101             }
102             if (r.overwrite)
103             {
104                 try {
105                     WebRequest request = HttpWebRequest.Create(baseUrl +
    "version.php?file=" + r.name);
106                     request.Timeout = 1000;
```

```
107                     HttpWebResponse response =
    (HttpWebResponse)request.GetResponse();
108                     var reader = new StreamReader(response.GetResponseStream());
109                     string result = reader.ReadToEnd();
110                     DateTime mt = DateTime.Parse(result);
111                     if (mt > System.IO.File.GetLastWriteTime(r.name))
112                     {
113                         return true;
114                     }
115                     else
116                     {
117                         return false;
118                     }
119                 } catch {
120                     return false;
121                 }
122             }
123             else
124             {
125                 return false;
126             }
127         }
128
129         private void Form1_Load(object sender, EventArgs e)
130         {
131             resources = new Resource[4];
132         resources[0] = new Resource("nova.jar", false, true); //Main Java Executable
133         resources[1] = new Resource("resources.zip", true, true); //Maps and static
    images
134         resources[2] = new Resource("settings.txt", false, false); //Default settings
    file
135             javaInstalled = getJavaVersion();
136             if (javaInstalled == JavaInstallStatus.NONE)
137             {
138                 MessageBox.Show("Java is not installed or broken. Please go to
    java.com and install (or reinstall) the latest available version.",
139                     "Fatal Error",
140                     MessageBoxButtons.OK,
141                     MessageBoxIcon.Error);
142                 Application.Exit();
143                 return;
144             }
145             t = new System.Timers.Timer(100);
146             t.Elapsed += new System.Timers.ElapsedEventHandler(runStuff);
147             t.Start();
148         }
149
150         private void runStuff(object source, System.Timers.ElapsedEventArgs e)
151         {
152             IntPtr ptr = new IntPtr();
153             Wow64DisableWow64FsRedirection(ref ptr);
154             t.Stop();
155             if (javaInstalled == JavaInstallStatus.JAVA_64)
156             {
157             resources[3] = new Resource("natives_x64.zip", true, true); //Native
    libraries, x64 version
158             }
```

```
159             else
160             {
161                 resources[3] = new Resource("natives_x86.zip", true, true); //Native libraries, x86 (32-bit) version
162             }
163
164             if (System.IO.File.Exists("installed.txt"))
165             {
166                 firstRun = false;
167             }
168             else
169             {
170                 firstRun = true;
171             }
172             Ping ping = new Ping();
173             try
174             {
175                 PingReply reply = ping.Send("davidstech.net", 1000);
176                 if (reply == null)
177                 {
178                     online = false;
179                 }
180                 else
181                 {
182                     if (reply.Status == IPStatus.Success)
183                     {
184                         online = true;
185                     }
186                     else
187                     {
188                         online = false;
189                     }
190                 }
191             }
192             catch
193             {
194                 online = false;
195             }
196
197             if ((!online) && firstRun)
198             {
199                 MessageBox.Show("Could not connect to remote server. Internet connectivity is needed for first run.",
200                     "Fatal Error",
201                     MessageBoxButtons.OK,
202                     MessageBoxIcon.Error);
203                 Application.Exit();
204                 return;
205             }
206             label1.Text = "Checking for updates...";
207             if (online)
208             {
209                 foreach (Resource r in resources)
210                 {
211                     if (needsInstall(r))
212                     {
213                         label1.Text = "Downloading " + r.name;
```

```
214                     WebClient client = new WebClient();
215                     try
216                     {
217                         Application.DoEvents();
218                         client.DownloadFile(baseUrl + r.name, r.name);
219                         Application.DoEvents();
220 
221                     }
222                     catch (WebException exception)
223                     {
224                         MessageBox.Show("Could not download " + r.name + " : " +
    exception.Message + ".",
225                             "Download Failed",
226                             MessageBoxButtons.OK,
227                             MessageBoxIcon.Exclamation);
228                         if (firstRun)
229                         {
230                             Application.Exit();
231                             return;
232                         }
233                     }
234                     catch
235                     {
236                         MessageBox.Show("Could not download " + r.name + ".",
237                             "Download Failed",
238                             MessageBoxButtons.OK,
239                             MessageBoxIcon.Exclamation);
240                         if (firstRun)
241                         {
242                             Application.Exit();
243                             return;
244                         }
245                     }
246                     if (r.extract)
247                     {
248                         label1.Text = "Extracting " + r.name;
249                         Application.DoEvents();
250 
251                         try
252                         {
253                             Application.DoEvents();
254                             ZipArchive z = ZipFile.OpenRead(r.name);
255                             foreach(ZipArchiveEntry za in z.Entries) {
256                                 try
257                                 {
258                                     System.IO.File.Delete(za.FullName);
259                                 }
260                                 catch
261                                 {
262 
263                                 }
264                             };
265                             ZipFile.ExtractToDirectory(r.name, ".");
266                             Application.DoEvents();
267 
268                         }
269                         catch(Exception ex)
```

```
270                     {
271                         MessageBox.Show(ex.ToString());
272                         MessageBox.Show("Could not extract " + r.name + ".",
273                               "Zip File Error",
274                               MessageBoxButtons.OK,
275                               MessageBoxIcon.Exclamation);
276                         if (firstRun)
277                         {
278                             Application.Exit();
279                             return;
280                         }
281 
282                     }
283                 }
284             }
285         }
286     };
287 
288     if (firstRun)
289     {
290         label1.Text = "Finished";
291         System.IO.File.Create("installed.txt").Close();
292         DialogResult r = MessageBox.Show("Would you like to create a desktop
    shortcut?",
293             "Mission Control Setup",
294             MessageBoxButtons.YesNo);
295         if (r == DialogResult.Yes)
296         {
297             object shDesktop = (object)"Desktop";
298             WshShell shell = new WshShell();
299             string shortcutAddress = (string)shell.SpecialFolders.Item(ref
    shDesktop) + @"\Mission Control.lnk";
300             IWshShortcut shortcut =
    (IWshShortcut)shell.CreateShortcut(shortcutAddress);
301             shortcut.Description = "Team Nova Mission Control";
302             shortcut.TargetPath =
    System.Reflection.Assembly.GetExecutingAssembly().Location;
303             shortcut.WorkingDirectory = Environment.CurrentDirectory;
304             shortcut.Save();
305         }
306     }
307     label1.Text = "Starting...";
308     var processInfo = new ProcessStartInfo("java.exe","-jar nova.jar")
309     {
310         CreateNoWindow = true,
311         UseShellExecute = false
312     };
313     Process p = Process.Start(processInfo);
314     System.Threading.Thread.Sleep(500);
315     SetForegroundWindow(p.MainWindowHandle);
316     Application.Exit();
317 
318         }
319     }
320 }
```