Sensors in Space

A science project by

Moonty1

Team Members:

Sarah Buckley,
Sean Murphy,
Dylan Halpin Hurley

(and Rachel Sheehan)
Supervising Teacher: L. Begley

Mayfield Community School
SciFest, CIT, March 2017

Contents

- Meet the Team
- Summary
- Introduction
- Background Research
- Experimental Methods
- Results
- Conclusions and Recommendations
- Acknowledgements
- References
- Appendices

Meet the team

We are transition year students in Mayfield Community School in Cork. In September, our teacher encouraged a number of us to enter a competition to develop an App for Android devices. We didn't complete our entries but when he told us about this ESA competition we decided to try our hand at coding again and complete it this time. We just recently learned that our entries for the two programming challenges has been successful and our code will be run on a Raspberry Pi computer on the International Space Station at the beginning of May.
We are:

- Sarah Buckley : Team leader, main job was to make sure everyone knew when and where to meet and what they were to do.
    - Also to help the other team members with their own jobs.

- Dylan Halpin Hurley : Main job was writing the code for the images that would appear on the LED Matrix.
    - Also helping to tidy up the code at the end.

- Sean Murphy : Main job was writing the code for the Bendyfield code.
    - Also checking form areas we could improve on.

For the AstroPi project we had four members on our team but we can only enter three names for SciFest. Rachel drew the short straw but this is what she did:

- Rachel Sheehan : Main job was deciding the colours for the images that would appear on the LED Matrix.
    - Also helping to tidy up the code at the end.

Summary

We started this project when we entered the Astro pi challenge that was run by the European Space Agency (ESA).
For this competition we had to write two different programs using the Python programming language.
The first part of the competition was to write code to detect movements in and out of the Columbus Module on the ISS. We did this by using the the humidity sensor on the AstroPi version of the Raspberry Pi. We called this program Astrobods.
The second part of this competition was to write a program to conduct an experiment of our choice. The experiment we decided to do was to write code to detect any anomalies in the earth's magnetic field such as the South Atlantic Anomaly. We call this program Bendyfield.
We completed this by using the magnetometer sensor on the Raspberry Pi, and also by using the orientation sensor and the accelerometer to cross-reference any anomalies detected, so as to make any data collected more accurate.
Once we were finished our code we sent it away to be judged and patiently waited for the results. A while later we found out that we were the Irish winners and our code would be going to the ISS.
We have developed an Excel worksheet to analyse the data that will be collected. It uses a Line Chart to present the Astrobod data and Radar Charts to present the 3_D data from Bendyfield.

Introduction

When we first heard this competition, we thought that it would be a great opportunity to, firstly, learn more about coding and secondly, to collect data on the subject of our experiment (which we later decided would be the earth's magnetic field) right from the source rather than analysing data that was collected by someone else.

We chose the magnetic field as the subject of our experiment as it has been discovered that it is not a straightforward pattern but has bumps and anomalies at certain places.

We thought that this would be a good choice of experiment as it was a once in a lifetime chance to be able to collect such data, that we would be able to analyse later if our code was chosen.

Also to get this code to work and to work accurately we would have to write an intricate code that would hopefully make us stand out from our competitors.

This competition had two parts. In the first part, everyone who entered had to do the same challenge, which was to detect movement in and out of the Columbus Module of the International Space Station (ISS).

This was the first piece of coding that some of us had ever done so was hard. Our teacher was very helpful and very patient. This code is called Astrobods and is organised into three files.

We used this code sort of as practice before moving on to the harder second part. We called this code Bendyfield and it is also organised into three files.

We should also add that some of us had previously entered an App development competition but didn't manage to complete it. This was a chance for Sean, Dylan and Rachel to try again and complete the course.

Background Research

One of the first things we had to research was Python Programing language, as three of us had never worked with coding before. We used Python 3.4.2 which is pre-loaded on the Raspberry Pi. After that we had to research the Raspberry Pi, how it works and the sensors that come with the additional attachment called the SenseHat.

The AstroPi was developed by the Raspberry Pi Foundation in the UK and the UK Space Agency. The astronaut Tim Peake brought two of these to the ISS and they are still there.

The thing we most had to research was the earth's magnetic field. For this we researched magnetic anomalies which was what our experiment was about.

We also had to research about how to code the accelerometer, gyroscope and magnetometer sensors to work the way we needed them to for our Bendyfield program.

The AstroPi website (www.astropi.org) was a great help with these tasks thanks to it's helpful videos and reference materials and also the Python tutor websites.

We found a lot of information on the NASA website and also the European Space Agency's website .

Experimental methods

Astrobods:

In the first part of the competition we had to detect crew movements in and out of the Columbus Module, we did this by using a humidity sensor to monitor crew entering and leaving the module by comparing humidity levels with baseline humidity.
We wrote our code in the programming language Python 3.4.2.
The code samples the humidity at 10 second intervals.
The code also re-samples the baseline humidity at regular intervals.
To make our code easy to read we split it into three parts, Astrobods_8a was the main program, Astrobods_8b was the functions file and Astrobods_8c was the file for the LED matrix images. This made the code easy to read as we were able to do all calculations in the functions file and just have the settings and main loop in the main program.
For the LED matrix images, we first had to design what image would appear on the matrix before coding it.
We also had to decide on small things like colour, we had to think about how the colours would look on ISS and how bright it would appear.
For this program we had to make multiple images for each possible outcome.
The image we picked was of a stickman with the number of the crew that had left or entered.

Bendyfield:

For the second part of the competition we chose to do our experiment on the earth's magnetic field. The ISS is in low earth orbit and so it should be able to detect anomalies such as the south atlantic anomaly.
Our experiment was to find any anomalies in the magnetic field.
To do this we used a magnetometer sensor to track 3-dimensional changes in the earth's magnetic field in microTeslas (μT) and cross-reference any anomalies against any changes in ISS orientation using the orientation sensor (pitch, roll and yaw measured in degrees), orientation towards geomagnetic north using the compass or against

changes detected by the accelerometer such as rocket boosters firing to adjust the ISS orbit.
While it does this it also takes note of the date and time.
Our main problem is that we will only get to run our code for 90 minutes, which is the time it takes for the ISS to orbit the earth once. We just hope we are lucky with the orbit direction when our code is run.
For this code we did the same as the first, splitting it into three parts.
We also wrote this code in Python 3.4.2.
The sampling time for this code is 1 second intervals for each of the sensors.
Bendyfield_8a was the main program, Bendyfield_8b was the functions file and Bendyfield_8c was the LED matrix images, this as it did in the first part makes the code easier to read and understand.
As this program wasn't an interactive one we only had one LED matrix image appear.

Program Output:

The LED display is the only visible output on the ISS but other messages are displayed on the Python window to show the progress of the program. These are visible on the monitor attached to our AstroPi on our display stand.

Results

As our code has yet to actually go up to the ISS.
we don't have any results from our actual experiment but we do have test data from when we were testing our code. We have also developed an excel sheet to analyse the data we get back in May.
Breathing on the Humidity sensor can be used to test the Astrobods program, either while the baseline is set, whereby you can end up with negative numbers indicating a crew member exit; or after the baseline is set, whereby you end up with positive numbers indicating a crew member entrance.
Our program allows for up to 8 crew members entering or leaving.
According to www.space.com, the most people to be on the ISS at any one time so far is thirteen, but that is only when the crew were joined by crew from the transport rocket that docked with them.
We tested Bendyfield by moving the SenseHat around to get gyroscope, compass, accelerometer and primarily magnetometer data changes and also a bar magnet to cause changes in the magnetic field.
The data file for Bendyfield has a wider mix of data than Astrobods but there was very little User Interface code to worry about.

There was quite a bit of difficult maths in getting the logic that controlled the code but our teacher helped a lot with that.

Examples of the test data and charts for Astrobods and Bendyfield are in the Appendix.

The Astrobods results are presented using a Line Chart.
The Bendyfield results are presented using three Radar Charts.

Conclusions

We now know that our code was chosen and we are the Irish winners. Our code will now be uplinked to the ISS to run on the Astro Pi's in the Columbus module, once the data is collected it will be downlinked to Earth and we will receive it by the 15th of May 2017.
We will then be able to analyse the data and conclude our experiment. What we can say at this point is that the code works with our test data under simulated conditions, but that the proof of the pudding will be in the data files that come back to us from space. It's kinda cool.

Acknowledgements

We would like to give our thanks to the European Space Agency for the help they gave to start us off on our program.
We would also like to thank our teacher Mr. Begley for teaching us how to code and all the other help he gave us.
We would like to thank our school for timetabling Computer Studies for us in Transition Year and for making the school computer and IT facilities available to us.
We would like to thank Mr. Eddie Higgins for loaning us two Raspberry Pi computers.
We would like to thank you for taking the time to read our project file and we hope you take the opportunity to try out the AstroPi computers and our Astrobods and Bendyfield programs while visiting our stand.

References

Some of the websites we used were

- astro-pi.org/resources/
- www.raspberrypi.org/learning/astro-pi-guide/
- http://www.esa.int
- www.python.org
- www.space.com

Appendices

1. ESA notification:

Dear Moonty1 Team,
Congratulations! Your code has been qualified to fly and run on the International Space Station (ISS).

ESA, in collaboration with the Raspberry Pi Foundation, would like to congratulate you all for your excellent work!

After receiving more than 180 entries, it was very difficult to choose the best experiments to run on the ISS. An evaluation panel, composed of ESA and the Raspberry Pi Foundation, selected the experiments that have been qualified to fly. The submissions were evaluated based on the following criteria: scientific value, innovation, feasibility of the mission within the ISS environment, clarity, comprehensiveness, code quality and readability. Please find below a short comment from the jury about your mission:

The judges were impressed by how well structured and commented your code was, great use of custom modules to keep everything neat and tidy. They were particularly amused by the where_is_santa function. Great work!
Your code will now be uplinked to the ISS to run on the Astro Pi's in the Columbus module, under the careful supervision of ESA astronaut Thomas Pesquet. The collected data will be downlinked to Earth and distributed to the teams by 15 May 2017. You will then have the chance to analyse the data collected in space and continue your amazing work – like a real space scientist!
Once again, we would like to congratulate all of the teams, students, and teachers, that participated in this project.

Kind Regards,
The ESA Education Team

2. Astrobods Code

a) Astrobods_9a.py

```
#############################################################################
# Moonty1                                                   #
#                                                           #
# The first Challenge!                                      #
#                                                           #
# Astronaut detector program: Astrobods                     #
#                                                           #
# This will record the mean background humidity of the Columbus module. #
# It will then monitor for changes in humidity.             #
# If the changes are more than 4% (i.e. more than one person exit/entry #
# then it records this.                                     #
# The mean background humidity is re-sampled at intervals.  #
# As the data is monitored, it is also stored in a log file #
#                                                           #
# This program will run for approximately 86 minutes        #
#   (85 minutes of crew monitoring, 1 minute for base humidity resets   #
#    and a margin for delays in input, output and CPU time)  #
#                                                           #
#############################################################################
# Team members:  Sarah Buckley, Dylan Halpin-Hurly, Rachel Sheehan,     #
#               Sean Murphy (and Mr Begley)                 #
#                                                           #
#############################################################################
#                                                           #
# version 8                                                 #
#                                                           #
#############################################################################
#                                                           #
#  The main program                                         #
#                                                           #
#############################################################################

#import libraries
from sense_hat import SenseHat
import datetime
from time import gmtime, strftime
import statistics
import math
import Astrobods_9b as astrof # import the functions file
import Astrobods_9c as astroi # import the LED matrix images

#
#  set aliases
#
sense = SenseHat()

##############################
#  Welcome to the program   #
#  En Francais: Bienvenue   #
#  As Gaeilge: Dia dhaobh   #
##############################
```

```
sense.set_rotation(270) # arrange for the messages to read the right way up in the ISS
#sense.show_message("Moonty1 from Cork: Astrobods",
#                   text_colour=[128, 128, 128],
#                   scroll_speed= 0.03) # say Hello!
# show splash image on the LED matrix in white text
astroi.show_splash()
print("Starting Astrobods crew-tracking program")

#######################################
# Program Settings                    #
#######################################
precision = 2 # Set the precision of the readings
baseline_time = 10 # number of seconds to use for baseline readings
sample_time = 10 # number of seconds in each loop that monitors humidity
one_astrobod = 0.04 # humidity change for one crew member i.e. 4%
astrobod_runs = int(86*60/10) # number of cycles for the program to run (less than 1.5 hours)
baseline_gap = 15*60/10 # the number of loops to do before recalculating the baseline
astrobod_change = 0 # initialise the number of crew members entering or leaving
log_line = [] # initialise the list of data we will send to the log file
output_string = "" # initialise the string that will contain the data
#######################################
# Log file settings                   #
#######################################
filename = "Astrobods.csv"
WRITE_FREQUENCY = 1
astrof.setup_log_file(filename, precision)
print( "Runs:", astrobod_runs, "Gap:", baseline_gap, astrobod_runs % baseline_gap)

#######################################
#  Main Loop                          #
#######################################
for timer in range( 0 , astrobod_runs ):
    print("Run number: ", timer+1)
    log_line = []  # reset the log_line data list
    output_line = [] # reset the output string
    #
    # note the date and time
    #
    timestamp = strftime("%d/%m/%Y %H:%M:%S", gmtime())
    log_line.append( timestamp ) # time stamp the data
    #
    # is it time for a new baseline reading and astrobod_change?
    #
    if ( (timer % baseline_gap) == 0): # use Modulo to decide when the baseline gap is reached
        print("Resetting the baseline humidity value...")
        #astroi.show_splash() # display the question mark until there is a crew change
        #astroi.show_plus_4()
        baseline_humidity = astrof.get_baseline_humidity(baseline_time, precision)
        astrobod_change = 0  # reset the change-of-crew-member count
    print("Baseline humidity: ", baseline_humidity, "mb")
    log_line.append(baseline_humidity)
    #
    #  Has someone entered or left?
    #
```

```
    #  1. Has the humidity changed significantly from the baseline humidity
    #     by at least 4% (or one_astrobod humidity value)
    #     over the number of seconds in the sample_time variable
    #     to the number of decimal places in the precision variable
    #
    humidity_change = astrof.monitor_humidity(baseline_humidity, sample_time, one_astrobod, precision)
    print("Humidity change over", sample_time, "seconds: ", humidity_change, "%")
    log_line.append(sample_time)  # timestamp the log line
    log_line.append(humidity_change) # record the % humidity change in the log line
    #
    # if the humidity change is more than + or - 4% then yes
    # if no other factors involved, each change of approximately 4% should be the same as
    # one crew member
    #
    #  2. How many have either come in or gone out?
    #
    if (humidity_change != 0): # if the humidity_change is not zero
        astrobod_change = round(humidity_change/(one_astrobod*100))
        # divide by one crew members humidity to see how many came in or out
    # -1 means one crew member has left, -2 means two have left, and so on
    # +1 means one crew member has entered, +2 means two have entered, and so on.
    print("Crew member changes in the module: ", astrobod_change )
    #
    # Set the LED matrix to show the crew change
    #
    #if (astrobod_change != 0):
    astroi.update_LED(astrobod_change)
    # put the astrobod_change in the output list
    log_line.append(astrobod_change)
    #print("Logline", log_line)
    # put the log_line data in a string
    output_string = astrof.log_string( log_line )
    print("output string", output_string)
    #
    # write the output_string to the log file
    #
    print("Writing to the log file \n")
    with open(filename,"a") as f:
        f.write( output_string  )

#
# Farewell / Adieu / Slan
#
#sense.show_message("Adieu from Moonty1 Astrobods",
#                   text_colour=[255, 255, 255],
#                   scroll_speed= 0.1) # say Hello!

# clear the LED
sense.clear()
```

b) Astrobods_9b.py

```
# Moonty1
# Astronaut detector program: Astrobods
#
# version 9
#
#  The functions file
#

#import libraries
from sense_hat import SenseHat
import time
import statistics
import math

#
#  set aliases
#
sense = SenseHat()


#
# This is a function to get and print the humidity, temperature and pressure.
def what_H_T_P(dp):
    returnstring = []
    humidity = round(sense.get_humidity(), dp ) # get the humidity reading
    temp = round(sense.get_temperature(), dp)  # get the temperature reading
    pressure = round(sense.get_pressure(), dp) # get the pressure reading
    #put the readings in a string message
    outputstring = "H:" + str(humidity) + "%rH, T:" + str(temp) + "C, P:" + str(pressure) + "mb"
    #print(outputstring)
    returnstring.extend( [humidity, temp, pressure] )
    return(outputstring)
    #return(returnstring)

#
# This is a function to return the baseline average humidity.
# This is calculated to the decimal precision
# and for the approximate length of time
# and returns the mean humidity
def get_baseline_humidity(sampletime1, dp):
    humidity = round(sense.get_humidity(), dp ) # get the humidity reading
    humiditysum = 0 # initialise the running total of humidity readings
    for i in range(0, sampletime1):
        humiditysum = round(humiditysum + humidity, dp)
        #print( i+1, "H: ", humidity, humiditysum) # print the humidity and running total
        time.sleep(1) # pause for 1 second
        humidity = round(sense.get_humidity(), dp ) # get the humidity reading
    mean_humidity = round(humiditysum/sampletime1, dp)
    #print("The average humidity over the last", sampletime, " seconds is: ", mean_humidity, "mb")
    return(mean_humidity)

#
# This is a function to monitor how the humidity is changing from the baseline.
```

```
# If it changes more than 4% than an astronaut may have entered or left the module.
#
def monitor_humidity(mean_hum, sampletime2, pc, dp):
    #humidity = round(sense.get_humidity(), precision ) # get the humidity reading
    humiditydiff = 0 # initialise the difference from the mean
    astrobod_alert = 0 # initialise the astrobod alert
    for i in range(0, sampletime2):
        humidity = round(sense.get_humidity(), dp) # get the humidity reading
        humidity_diff_int = round(humidity - mean_hum, dp)  # store difference in humidity_diff_int
        humidity_diff_pc = round(humidity_diff_int/mean_hum*100, dp) # store % difference
        if (abs(humidity_diff_pc) > pc) and (abs(humidity_diff_pc) > abs(astrobod_alert)) :
            # if humidity difference more than one body's humidity and higher than a previous alert
            astrobod_alert = humidity_diff_pc
        # print humidity and difference
        # print( i+1, "H: ", humidity, "delta: ", humidity_diff_int, "mb ", humidity_diff_pc, "%")
        time.sleep(1) # pause for one second
    return (astrobod_alert)

#
# This is a function to write the data in our log line to a comma
# delimited string
#
def log_string( data_line ):
    comma_string = ",".join(str(value) for value in data_line)
    comma_string = comma_string + "\n"
    #print("comma_string:", comma_string)
    return(comma_string)

#
# This function sets up the log file
#
def setup_log_file( filename, dp ):
    title_1 = "Moonty1, Proxima, Astrobods, log, file \n"
    title_2 = what_H_T_P( dp )
    headings =  [
                "TimeStamp", "Base_H", "S_time", "H_Diff", "Crew_diff"
                ]
    units =     [
                "DMY H:M:S","%rH","s","%","heads"
                ]
    with open(filename,"w") as f:
        f.write(title_1)
        f.write(",".join(str(value) for value in headings) + "\n")
        f.write(",".join(str(value) for value in units) + "\n")
```

c) Astrobods_9c.py

```
# Moonty1
# Astronaut detector program: Astrobods
#
# version 8c
#
#  LED images for Astrobods
#

#import libraries
from sense_hat import SenseHat
import time
import math

#
#  set aliases
#

sense = SenseHat()

#
#  This is our splash screen
def show_splash():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    O = [0, 0, 0]  # Black

    question_mark = [
    O, O, G, G, O, O, O, O,
    O, G, O, O, G, G, O, G,
    O, O, G, O, O, O, O, O,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(question_mark)

#
#  This is our zero screen
def show_no_change():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    O = [0, 0, 0]  # Black

    no_change = [
    O, O, O, O, O, O, O, O,
    O, O, O, O, O, O, O, O,
    O, O, O, O, O, O, O, O,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
```

```
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(no_change)
#
#  This is our plus 1 screen
def show_plus_1():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    plusone = [
    O, P, O, O, O, O, O, O,
    P, P, P, G, G, G, G, G,
    O, P, O, O, G, O, O, O,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(plusone)


#
#  This is our plus 2 screen
def show_plus_2():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    plustwo = [
    O, P, O, O, G, O, O, G,
    P, P, P, G, O, G, O, G,
    O, P, O, G, O, O, G, G,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(plustwo)

#
#  This is our plus 3 screen
def show_plus_3():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    plusthree = [
    O, P, O, O, G, O, G, O,
```

```
    P, P, P, G, O, G, O, G,
    O, P, O, G, O, O, O, G,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(plusthree)

#
#  This is our plus 4 screen
def show_plus_4():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    plusfour = [
    O, P, O, O, O, G, O, O,
    P, P, P, G, G, G, G, G,
    O, P, O, O, O, G, O, O,
    O, O, W, G, G, G, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(plusfour)


#
#  This is our plus 5 screen
def show_plus_5():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    plusfive = [
    O, P, O, G, O, G, G, G,
    P, P, P, G, O, G, O, G,
    O, P, O, G, G, G, O, G,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(plusfive)

#
#  This is our plus 6 screen
def show_plus_6():
```

```
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    plussix = [
    O, P, O, G, O, G, G, G,
    P, P, P, G, O, G, O, G,
    O, P, O, G, G, G, G, G,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(plussix)

#
#  This is our plus 7 screen
def show_plus_7():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    plusseven = [
    O, P, O, G, G, G, G, G,
    P, P, P, G, O, O, O, O,
    O, P, O, G, O, O, O, O,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(plusseven)


#
#  This is our plus 8 screen
def show_plus_8():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    pluseight = [
    O, P, O, G, G, G, G, G,
    P, P, P, G, O, G, O, G,
    O, P, O, G, G, G, G, G,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
```

```
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(pluseight)



#
#  This is our minus 1 screen
def show_minus_1():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    minusone = [
    O, P, O, O, O, O, O, O,
    O, P, O, G, G, G, G, G,
    O, P, O, O, G, O, O, O,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(minusone)

#
#  This is our minus 2 screen
def show_minus_2():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    minustwo = [
    O, P, O, O, G, O, O, G,
    O, P, O, G, O, G, O, G,
    O, P, O, G, O, O, G, G,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(minustwo)

#
#  This is our minus 3 screen
def show_minus_3():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black
```

```
    minusthree = [
    O, P, O, O, G, O, G, O,
    O, P, O, G, O, G, O, G,
    O, P, O, G, O, O, O, G,
    O, O, O, O, W, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, O, O, W, O, O, O
    ]
    sense.set_pixels(minusthree)

#
#  This is our minus 4 screen
def show_minus_4():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    minusfour = [
    O, P, O, O, O, G, O, O,
    O, P, O, G, G, G, G, G,
    O, P, O, O, O, G, O, O,
    O, O, W, G, G, G, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(minusfour)

#
#  This is our minus 5 screen
def show_minus_5():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    minusfive = [
    O, P, O, G, O, G, G, G,
    P, P, P, G, O, G, O, G,
    O, P, O, G, G, G, O, G,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(minusfive)


#
```

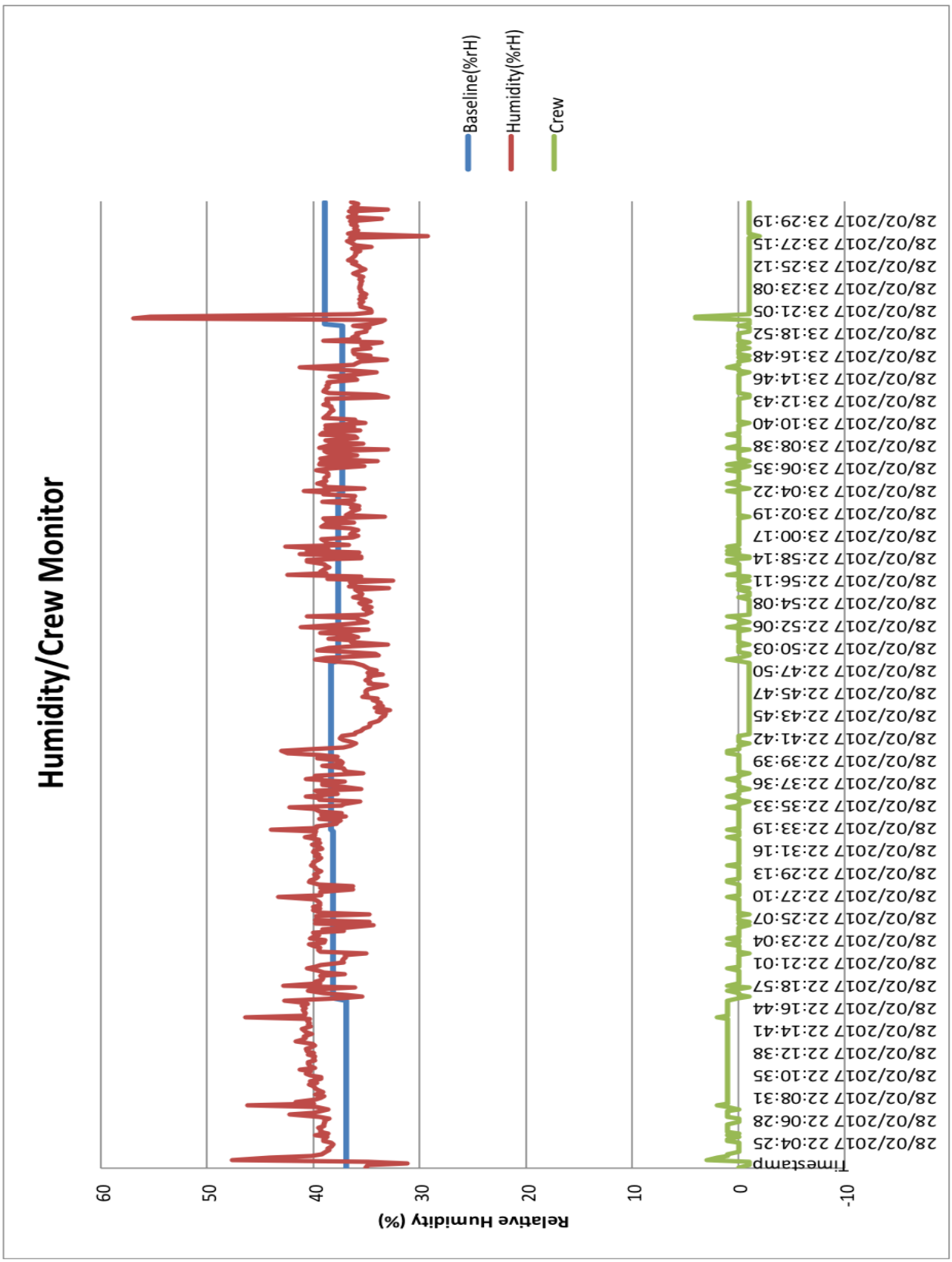
```
#  This is our minus 6 screen
def show_minus_6():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    minussix = [
    O, P, O, G, O, G, G, G,
    0, P, O, G, O, G, O, G,
    O, P, O, G, G, G, G, G,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(minussix)

#
#  This is our minus 7 screen
def show_minus_7():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    minusseven = [
    O, P, O, G, G, G, G, G,
    O, P, O, G, O, O, O, O,
    O, P, O, G, O, O, O, O,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(minusseven)

#
#  This is our minus 8 screen
def show_minus_8():
    G = [0, 255, 0]  # Green
    W = [255, 255, 255] # White
    P = [255, 0, 255] # Purple
    O = [0, 0, 0]  # Black

    minuseight = [
    O, P, O, G, G, G, G, G,
    O, P, O, G, O, G, O, G,
    O, P, O, G, G, G, G, G,
    O, O, W, O, O, O, O, O,
    W, W, O, W, O, O, W, W,
    W, W, W, W, W, W, O, O,
```

```
    W, W, O, W, O, O, W, W,
    O, O, W, O, O, O, O, O
    ]
    sense.set_pixels(minuseight)



#
# This is a function to update the LED matrix to show a crew exit/entry
#
def update_LED( change ):
    if (change == 0):
        show_no_change()
    elif (change == 1):
        show_plus_1()
    elif (change == 2):
        show_plus_2()
    elif (change == 3):
        show_plus_3()
    elif (change == 4):
        show_plus_4()
    elif (change == 5):
        show_plus_5()
    elif (change == 6):
        show_plus_6()
    elif (change == 7):
        show_plus_7()
    elif (change == 8):
        show_plus_-1()
    elif (change == -1):
        show_minus_1()
    elif (change == -2):
        show_minus_2()
    elif (change == -3):
        show_minus_3()
    elif (change == -4):
        show_minus_4()
    elif (change == -5):
        show_minus_5()
    elif (change == -6):
        show_minus_6()
    elif (change == -7):
        show_minus_7()
    elif (change == -8):
        show_minus_8()
    else:
        show_splash()
```

3. Astrobods Test Data

Timestamp	Baseline(%rH)	Humidity(%rH)	Crew	Humidity Difference(%)
28/02/2017 22:02:22	36.91	35.01	0	-1.9
28/02/2017 22:02:42	36.91	34.88	-1	-2.03
28/02/2017 22:02:53	36.91	31.17	-1	-5.74
28/02/2017 22:03:03	36.91	34.88	-1	-2.03
28/02/2017 22:03:13	36.91	47.61	3	10.7
28/02/2017 22:03:23	36.91	44.14	2	7.23
28/02/2017 22:03:34	36.91	40.16	1	3.25
28/02/2017 22:03:44	36.91	39.24	1	2.33
28/02/2017 22:03:54	36.91	38.78	0	1.87
28/02/2017 22:04:04	36.91	38.56	0	1.65
28/02/2017 22:04:15	36.91	38.64	0	1.73
28/02/2017 22:04:25	36.91	38.35	0	1.44
28/02/2017 22:04:35	36.91	38.16	0	1.25
28/02/2017 22:04:46	36.91	38.35	0	1.44
28/02/2017 22:04:56	36.91	39.02	1	2.11
28/02/2017 22:05:06	36.91	38.97	1	2.06
28/02/2017 22:05:16	36.91	38.7	0	1.79
28/02/2017 22:05:27	36.91	39.75	1	2.84
28/02/2017 22:05:37	36.91	38.59	0	1.68
28/02/2017 22:05:47	36.91	39.35	1	2.44
28/02/2017 22:05:57	36.91	39.35	1	2.44
28/02/2017 22:06:08	36.91	39.46	1	2.55
28/02/2017 22:06:18	36.91	38.97	1	2.06
28/02/2017 22:06:28	36.91	39.02	1	2.11
28/02/2017 22:06:38	36.91	38.89	0	1.98
28/02/2017 22:06:49	36.91	38.91	0	2
28/02/2017 22:06:59	36.91	38.51	0	1.6

4. Astrobods Chart



5. Bendyfield Code

a) Bendyfield_9a.py

```
###############################################################################
# Moonty1                                                                     #
#                                                                             #
# The second Challenge!                                                       #
#                                                                             #
# Magnetic Field Direction program: Bendyfield                                #
#                                                                             #
# This will record the direction of the magnetic field and save it in         #
# a .csv file to be analysed back on earth.  We want to see how it            #
# varies as the ISS orbits around the earth, if we have enough time.          #
#                                                                             #
# It will also monitor changes in the orientation and movement of the         #
# AstroPi and record them with the magnetic field data.  It will also         #
# monitor any changes in the direction to Geo Magnetic North and record       #
# these too.  We will see if any magnetic variations match changes in         #
# these measurements in the log file.                                         #
#                                                                             #
# This program will run for approximately 90 minutes                          #
#                                                                             #
###############################################################################
# Team members:  Sarah Buckley, Dylan Halpin-Hurly, Rachel Sheehan,            #
#                Sean Murphy (and Mr Begley)                                  #
#                                                                             #
###############################################################################
#                                                                             #
# Version 9                                                                   #
# This version has been updated slightly to compete at SciFest 2017           #
# at CIT on 31st March 2017 by adding more comments.                          #
#                                                                             #
# Version 7 won a place on the Proxima Mission and will be run on the         #
# on the ISS in May 2017.                                                     #
#                                                                             #
#                                                                             #
###############################################################################
#                                                                             #
#  The main program                                                           #
#                                                                             #
###############################################################################

#import libraries
from sense_hat import SenseHat
#from datetime import datetime
from time import gmtime, strftime
import time
import Bendyfield_9b as bendyf
import Bendyfield_9c as bendyi

#
```

```
#  set aliases
#
sense = SenseHat()

##############################
#  Welcome to the program   #
#  En Francais: Bienvenue   #
#  As Gaeilge: Dia dhaobh   #
##############################
sense.set_rotation(270) # arrange for the messages to read the right way up in the ISS
#sense.show_message("Moonty1 from Cork: Bendyfield",
#                   text_colour=[128, 128, 128],
#                   scroll_speed= 0.03)
bendyi.show_splash()  # show splash image on the LED matrix in white text

##################################
# Program Settings               #
##################################
precision = 2 # Set the number of decimal places we want
dir_err_margin = 1 # allow for a margin of error in difference in pitch, roll & yaw
runtime = 60*85 # the number of seconds to run the main loop for
log_line = [] # Initialise the  variable that we will send to the log file
output_string = "" # initialise the string that will contain the data
#####################################
# Log file settings                 #
#####################################
filename = "Bendyfield.csv"
WRITE_FREQUENCY = 1
bendyf.setup_log_file(filename)
#
# Which way to the Magnetic North Pole? :)
# (initialise the Geo North direction)
northpole = bendyf.where_is_santa(precision)
print("North: ", northpole)
#
# What is the orientation?
# ( this is so that we can check to see if a change in the magnetic field
#   matches a change in the orientation of the ISS )
# Initialise the orientation
b_pry = bendyf.get_P_R_Y(1) # get the initial P, R and Y directions
print("Orientation - b_pry:", b_pry)
same_way_up = 'yes' # Initialise the orientation change

#################
# Main Loop     #
#################
for count in range( 0, runtime):
    #
    # Reset the line that logs the data
    log_line = []
    # Is Geo North still in the same direction?
    # (will this value change as the orbit position changes?)
    check_north = bendyf.where_is_santa(precision)
    if (check_north != northpole): # Is the new reading the same as the base reading
```

```
        same_santa = 'no' # If the base reading is not the same, 'no' appears
        northpole = check_north # Make the new reading the base reading
    else:
        same_santa = 'yes' # If they are equal 'yes' will appear
    log_line.append(same_santa) # Record if it has changed or not
    log_line.append(northpole) # Record the north pole direction
    #
    # Has the orientation changed?
    check_pry = bendyf.get_P_R_Y(dir_err_margin)
    if (check_pry != b_pry): # Is the new reading the same as the base reading
        same_way_up = 'no' # If the base reading is not the same, 'no' appears
        b_pry = check_pry # Make the new reading the base reading
    else:
        same_way_up = 'yes' # If they are equal, 'yes' appears
    log_line.append(same_way_up) # Record if it has changed
    pitch = b_pry[0]
    roll  = b_pry[1]
    yaw   = b_pry[2]
    log_line.extend( [ pitch, roll, yaw ]) # Record the orientation
    #
    # What does the Accelerometer say?
    # ( this is so that we can check to see if a change in the magnetic field
    #   matches any change in the accelerometer on the ISS such as an orbit boost)
    a_xyz = bendyf.get_the_shakes(precision)
    a_x = a_xyz[0]
    a_y = a_xyz[1]
    a_z = a_xyz[2]
    log_line.extend( [a_x, a_y, a_z] ) # Record the accelerometer
    #
    # What does the magnetometer say?
    # This is the data we want to analyse later
    magnetic_field = bendyf.magnetic_field(precision)
    m_x = magnetic_field[0]
    m_y = magnetic_field[1]
    m_z = magnetic_field[2]
    #
    timestamp = strftime("%d/%m/%Y %H:%M:%S", gmtime()) # get the date and time
    log_line.append( timestamp ) # time stamp the data
    log_line.extend( [ m_x, m_y, m_z ] ) # Record the magnetometer
    print(count+1, log_line)
    #
    # Write this line of data to the log file
    print("Writing to the log file \n")
    with open(filename,"a") as f:
        f.write(",".join(str(value) for value in log_line) + "\n")
    #
    # pause for a second
    time.sleep(1)

# clear the LED
sense.clear()
```

b) Bendyfield_9b.py

```
################################################
# Moonty1                                      #
# Magnetic Field Direction program: Bendyfield #
#                                              #
# version 9                                    #
#                                              #
#  The functions file                          #
################################################

#import libraries
from sense_hat import SenseHat
#import time
#import statistics
#import math

#
#  set aliases
#
sense = SenseHat()


#
# This is a function to get the direction of Magnetic North
def where_is_santa( dp ):
    santa = round(sense.get_compass(), dp)
#    print("Magnetic North is at ", santa, "degrees")
    return(santa)

#
# This is a function to reduce the accuracy of the orientation data
def get_P_R_Y( margin ):
    pry = sense.get_gyroscope() # get gyro data without the magnetometer or accelerometer
    p = round(pry["pitch"] / margin)  # build in margin of error
    r = round(pry["roll"] / margin)  # build in margin of error
    y = round(pry["yaw"] / margin )  # build in margin of error
    pry_rounded = []
    pry_rounded.extend( [p, r, y] ) # put rounded readings in a list
    #print(" in def: ", pry_rounded)
    return(pry_rounded) # return the list of rounded readings

#
# This is a function to get the accelerometer data in Gs on each axis
def get_the_shakes( dp ):
    accel = sense.get_accelerometer_raw()
    a_x = round(accel["x"], dp)  # round off the x reading
    a_y = round(accel["y"], dp)  # round off the y reading
    a_z = round(accel["z"], dp)  # round off the z reading
    accel_r = []
    accel_r.extend( [a_x, a_y, a_z]) # store the rounded readings in a list
#    print("accelerometer: x=", a_x, "y=", a_y, "z=", a_z)
    return(accel_r)  # return the rounded list
```

```
#
# This is a function to get the magnetometer data
def magnetic_field( dp ):
    mag_field1 = sense.get_compass_raw()
    m_x = round(mag_field1["x"], dp)  # round off the x reading
    m_y = round(mag_field1["y"], dp)  # round off the y reading
    m_z = round(mag_field1["z"], dp)  # round off the z reading
    mag_field2 = []
    mag_field2.extend( [m_x, m_y, m_z]) # store the rounded readings in a list
#    print("Magnetic Field in microteslas: x=", m_x, "y=", m_y, "z=", m_z)
    return(mag_field2)  # return the rounded list

#
# Set up the log file and headings
def setup_log_file(logfile):
    title_1 = "Moonty1:, Proxima:, Bendyfield, log, file \n\n"
    header = [
        "SameNorth","GeoNorth",
        "SameWayUp","Pitch","Roll","Yaw",
        "a_X","a_Y","a_Z",
        "Time",
        "m_X","m_Y","m_Z"
        ]
    units = [
        "yes/no","degrees",
        "yes/no","degrees","degrees","degrees",
        "G","G","G",
        "DMY H:M:S",
        "microTeslas","microTeslas","microTeslas"
        ]

    with open( logfile , "w") as f:
        f.write(title_1)
        f.write(",".join(str(value) for value in header) + "\n")
        f.write(",".join(str(value) for value in units) + "\n")
```

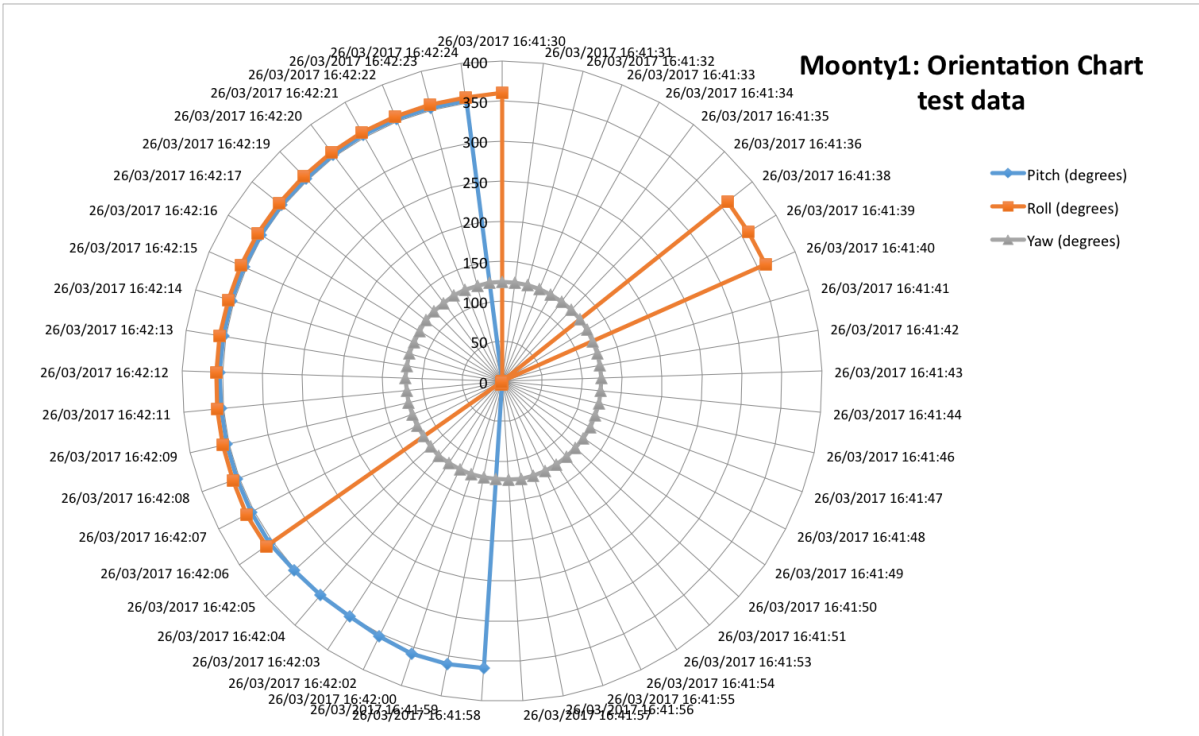
c) Bendyfield_9c.py

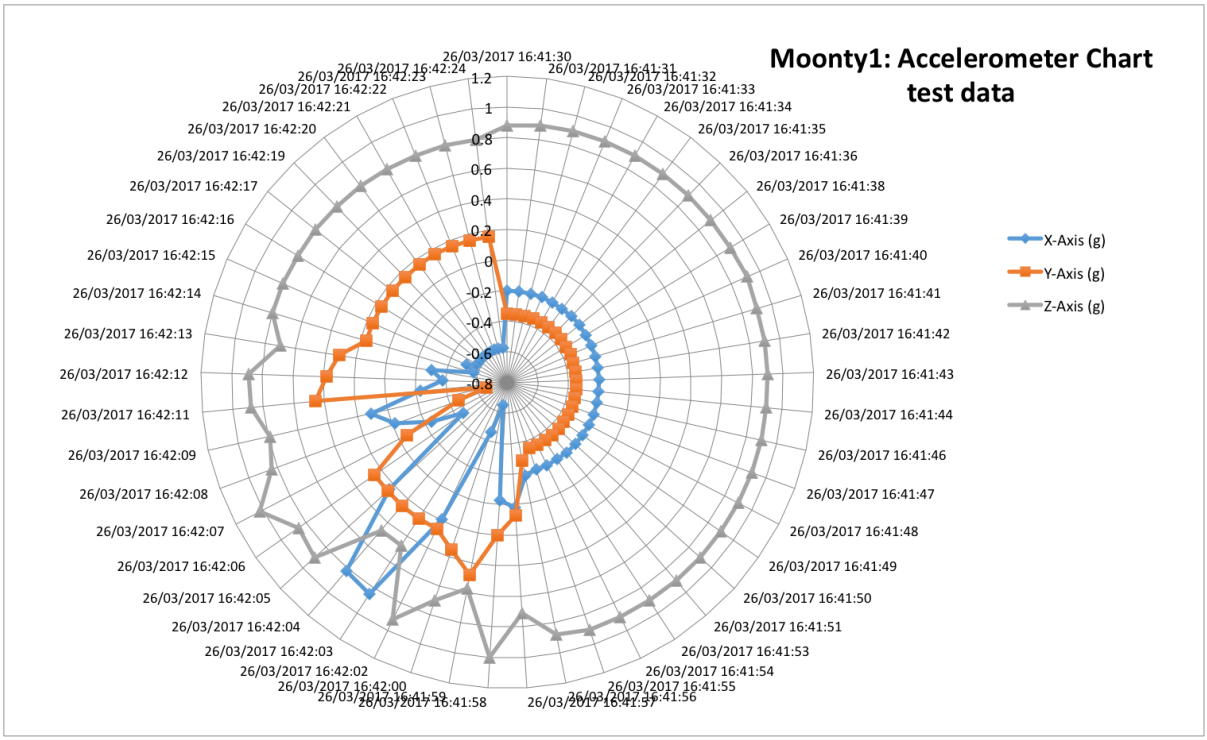
```
################################################
# Moonty1                                      #
# Magnetic Field Direction program: Bendyfield #
#                                              #
# version 9                                    #
#                                              #
#  LED images for Bendyfield                   #
################################################

#import libraries
from sense_hat import SenseHat
#import time
#import statistics
#import math

#
#  set aliases
#
sense = SenseHat()

#
#  This is our splash screen
def show_splash():
    G = [0, 128, 0]  # Green
    W = [128, 128, 128] # White
    O = [0, 0, 0]  # Black

    splash_screen = [
    G, O, O, G, O, O, O, O,
    O, G, O, O, G, O, O, O,
    O, O, G, G, G, G, G, G,
    O, O, O, W, G, O, O, O,
    O, O, O, G, W, O, O, O,
    O, O, W, W, W, W, W, W,
    O, W, O, O, W, O, O, O,
    W, O, O, W, O, O, O, O
    ]
    sense.set_pixels(splash_screen)
```

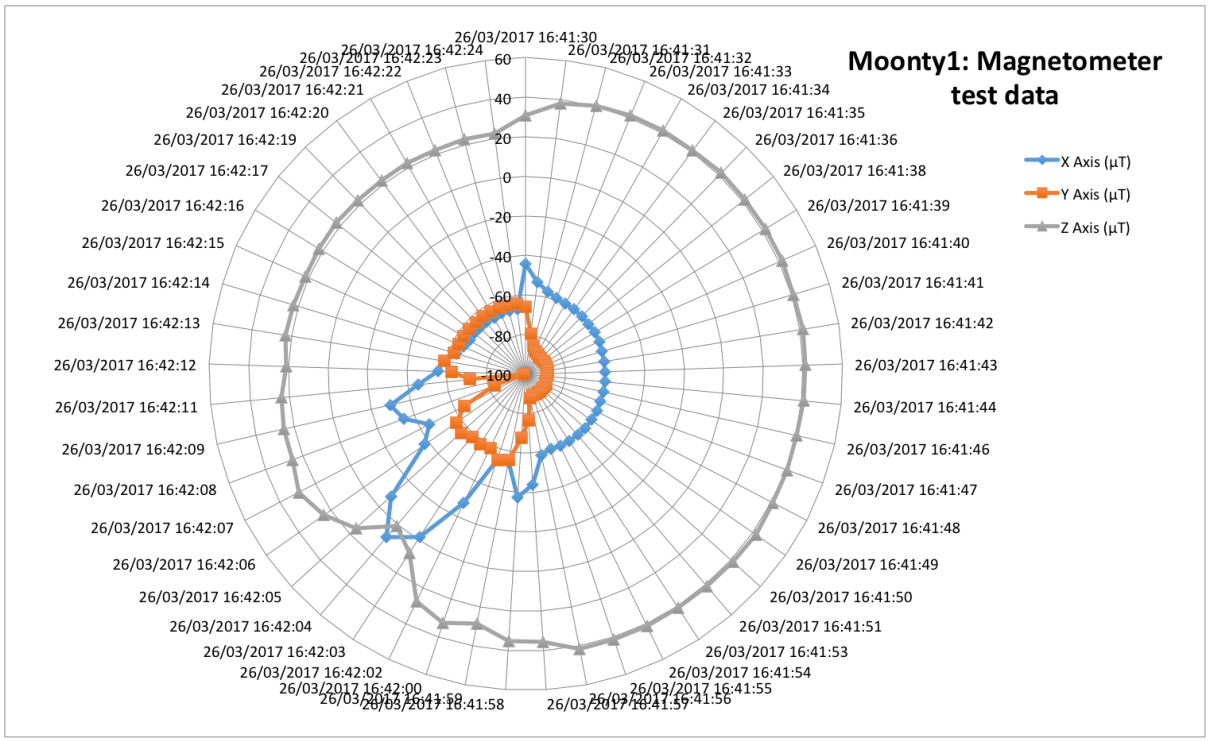
6. Bendyfield Test Data

SameNorth	GeoNorth	SameWayUp	Pitch	Roll	Yaw	a_X	a_Y	a_Z	Time	m_X	m_Y	m_Z
yes/no	degrees	yes/no	degrees	degrees	degrees	g	g	g	DMY H:M:S	microTeslas	microTeslas	microTeslas
no	123.74	yes	0	360	124	-0.2	-0.35	0.88	26/03/2017 16:41:30	-44.15	-65.82	30.78
no	123.73	no	0	0	124	-0.2	-0.35	0.89	26/03/2017 16:41:31	-53.27	-79.61	37.85
yes	123.73	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:32	-57.09	-85.17	40.27
yes	123.73	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:33	-58.71	-87.5	41.07
yes	123.73	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:34	-59.26	-88.34	41.35
yes	123.73	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:35	-59.27	-89.1	41.01
no	123.72	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:36	-59.55	-89.03	41.68
yes	123.72	no	0	360	124	-0.2	-0.35	0.9	26/03/2017 16:41:38	-59.49	-89.03	41.62
no	123.71	yes	0	360	124	-0.2	-0.35	0.9	26/03/2017 16:41:39	-59.2	-88.72	41.67
yes	123.71	yes	0	360	124	-0.2	-0.35	0.91	26/03/2017 16:41:40	-59.47	-88.91	41.58
no	123.7	no	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:41	-59.85	-88.81	41.17
yes	123.7	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:42	-59.75	-88.76	41.73
yes	123.7	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:43	-59.93	-89.05	41.35
yes	123.7	yes	0	0	124	-0.2	-0.35	0.9	26/03/2017 16:41:44	-59.75	-89.22	40.92

7. Bendyfield Charts






Moonty1: Magnetometer test data
X Axis (µT)
Y Axis (µT)
Z Axis (µT)
60
40
20
0
-20
-40
-60
-80
-100
26/03/2017 16:41:30
26/03/2017 16:41:31
26/03/2017 16:41:32
26/03/2017 16:41:33
26/03/2017 16:41:34
26/03/2017 16:41:35
26/03/2017 16:41:36
26/03/2017 16:41:38
26/03/2017 16:41:39
26/03/2017 16:41:40
26/03/2017 16:41:41
26/03/2017 16:41:42
26/03/2017 16:41:43
26/03/2017 16:41:44
26/03/2017 16:41:46
26/03/2017 16:41:47
26/03/2017 16:41:48
26/03/2017 16:41:49
26/03/2017 16:41:50
26/03/2017 16:41:51
26/03/2017 16:41:53
26/03/2017 16:41:54
26/03/2017 16:41:55
26/03/2017 16:41:56
26/03/2017 16:41:58
26/03/2017 16:42:00
26/03/2017 16:42:02
26/03/2017 16:42:03
26/03/2017 16:42:04
26/03/2017 16:42:05
26/03/2017 16:42:06
26/03/2017 16:42:07
26/03/2017 16:42:08
26/03/2017 16:42:09
26/03/2017 16:42:11
26/03/2017 16:42:12
26/03/2017 16:42:13
26/03/2017 16:42:14
26/03/2017 16:42:15
26/03/2017 16:42:16
26/03/2017 16:42:17
26/03/2017 16:42:19
26/03/2017 16:42:20
26/03/2017 16:42:21
26/03/2017 16:42:22
26/03/2017 16:42:23
26/03/2017 16:42:24