Alex Shah

Project 10 - Quad Copter downloading Roll+Pitch+Yaw, Lat+Lon+Alt and Video from RPi to Host via WiFi

2024-05-02

EN.605.715.81.SP24

Table of Contents

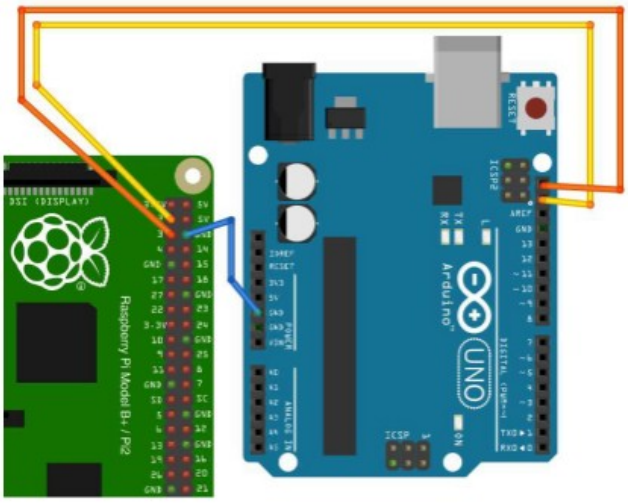
Requirements

We need to create a flight capable drone, put a Raspberry Pi on it, connect a usb camera and gps, and connect an arduino with an orientation sensor on it to the Pi and use I2C to communicate IMU data as well as GPS data and the camera feed to a remote machine over WiFi.

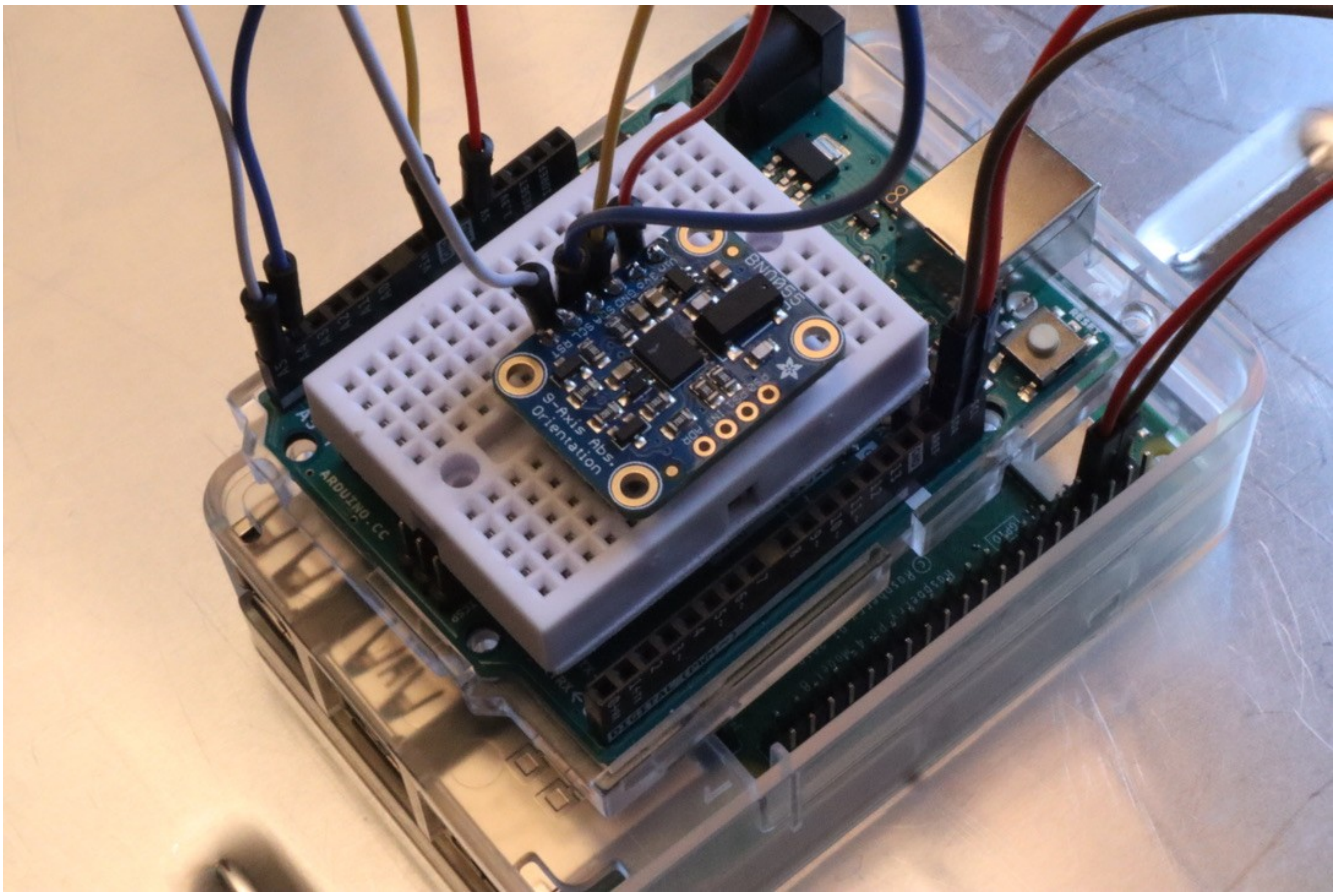
Design

Putting the Raspberry Pi 4, Arduino Uno, Adafruit BNO055 on a breadboard, gps sensor, and camera together I created a sensor stack that sits on top of the drone. I secured a battery pack to the bottom of the drone which doubled as a ballast I could slide around to achieve a better balance for stable flight.

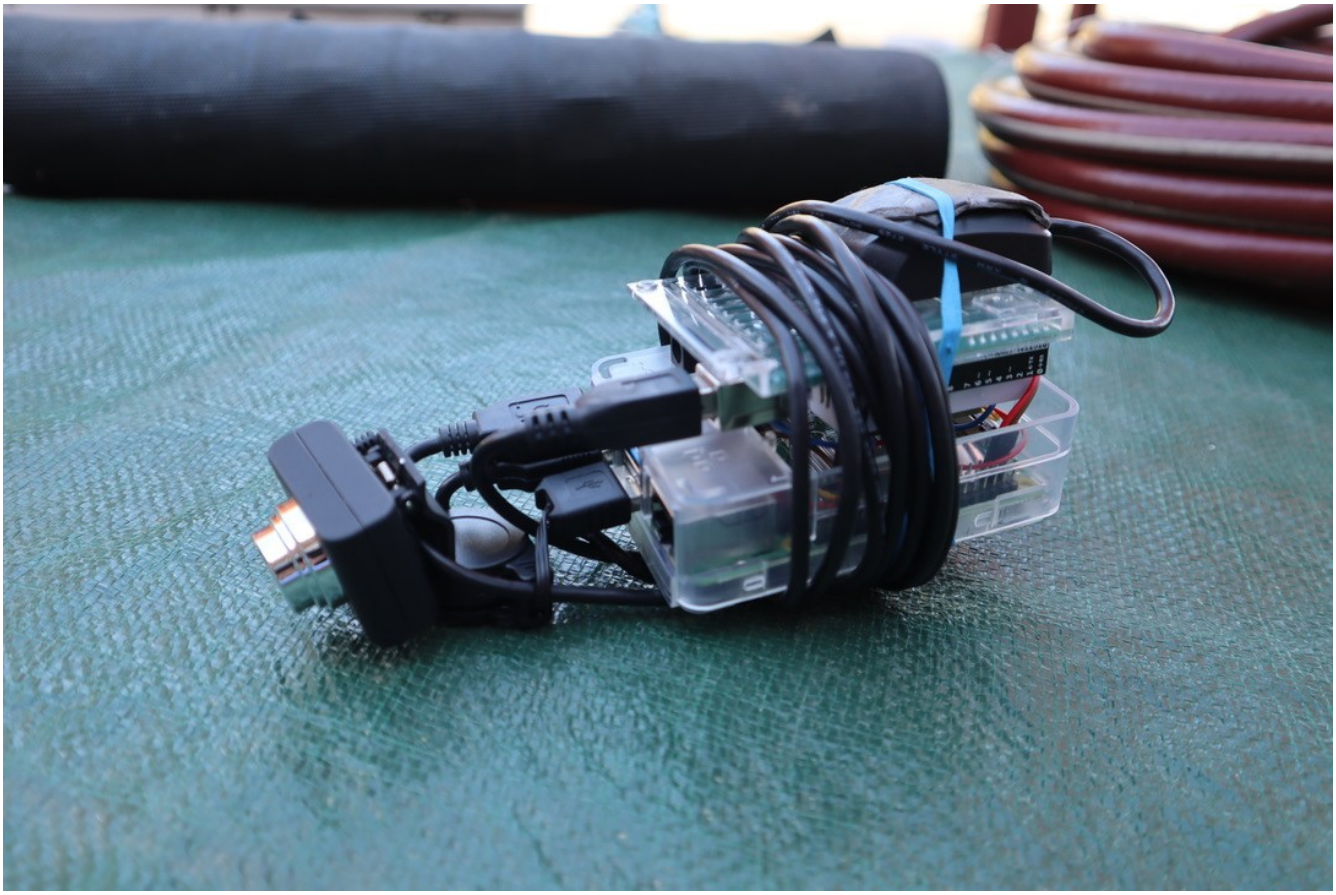
The Raspberry Pi 4 GPIO pins 2 and 3 (SDA and SCL) are connected to the digital pins on the top right of the Arduino UNO, the SDA and SCL pins, and GND to GND.



The Adafruit BNO055 is on a breadboard with wiring to connect to the Arduino with Vin to 5v, GND to GND, SDA to A4, and SCL to A5.



The breadboard with the sensor is stack onto the Arduino, which is stacked on to the Pi, then we connect the USB camera and USB GPS sensor to the Pi’s USB ports. This enables us to receive data from the sensors and camera, the gps and IMU data can be seen on /dev/ttyACM0 and /dev/tty/ACM1.



Implementation

A majority of the project has been built already, from the drone to the sensors and connections. However, we want to use I2C in order to send data from the arduino to the Pi, so there are some steps to enable this.

On the pi we need to enable I2C

$ sudo raspi-config

Choose to enable the I2C interface, then install dependencies

$ sudo apt install i2c-tools gpsd gpsd-clients python3-smbus

Later versions of python may not support smbus, so we can use

$ pip install smbus2

On the Arduino we need to capture IMU data from the sensor and prepare it to be sent over a wire. We use the Wire, Adafruit BNO055, and Adafruit Sensor libraries imported in Arduino IDE.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

/* This driver uses the Adafruit unified sensor library (Adafruit_Sensor),
   which provides a common 'type' for sensor data and some helper functions.

   To use this driver you will also need to download the Adafruit_Sensor
   library and include it in your libraries folder.

   You should also assign a unique ID to this sensor for use with
   the Adafruit Sensor API so that you can identify this particular
   sensor in any data logs, etc.  To assign a unique ID, simply
   provide an appropriate value in the constructor below (12345
   is used by default in this example).

   Connections
   ===========
   Connect SCL to analog 5
   Connect SDA to analog 4
   Connect VDD to 3-5V DC
   Connect GROUND to common ground

   History
   =======
   2015/MAR/03  - First release (KTOWN)
   2015/AUG/27  - Added calibration and system status helpers

```

```
30    @Author Modified by Addison Sears-Collins
31    @Date   April 17, 2019
32  */
33
34  /* Set the delay between fresh samples */
35  #define BNO055_SAMPLERATE_DELAY_MS (100)
36
37  Adafruit_BNO055 bno = Adafruit_BNO055(55, 0x28, &Wire);
38
39  // Flag used to stop the program
40  bool done = false;
41
42  // Make the Arduino a slave to the Raspberry Pi
43  int SLAVE_ADDRESS = 0X04;
44
45  // Toggle in-built LED for verifying program is working
46  int ledPin = 13;
47
48  // Data to send back to Raspberry Pi
49  byte imu_data[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
50
51  // Variables used for digit extraction
52  int roll = 0;
53  int pitch = 0;
54  int yaw = 0;
55
56  // Initialize the LED. This is used for testing.
57  boolean ledOn = false;
58
59  /*****************************************************************************/
```

```
60 /*
61    This function ends the program
62 */
63 /**************************************************************************/
64
65 void end_program() {
66
67    // Used for reading data from the serial monitor
68    char ch;
69
70    // Check to see if ! is available to be read
71    if (Serial.available()) {
72
73    // Read the character
74    ch = Serial.read();
75
76    // End the program if exclamation point is entered in the serial monitor
77    if (ch == '!') {
78      done = true;
79      Serial.println("Finished recording Roll+Pitch+Yaw data. Goodbye.");
80    }
81  }
82 }
83
84 /**************************************************************************/
85 /*
86    Displays some basic information on this sensor from the unified
87    sensor API sensor_t type (see Adafruit_Sensor for more information)
88 */
89 /**************************************************************************/
```

```
90 void displaySensorDetails(void)
91 {
92   sensor_t sensor;
93   bno.getSensor(&sensor);
94   Serial.println("------------------------------------");
95   Serial.print  ("Sensor:       "); Serial.println(sensor.name);
96   Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);
97   Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor_id);
98   Serial.print  ("Max Value:    "); Serial.print(sensor.max_value); Serial.println(" xxx");
99   Serial.print  ("Min Value:    "); Serial.print(sensor.min_value); Serial.println(" xxx");
100   Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" xxx");
101   Serial.println("------------------------------------");
102   Serial.println("");
103   delay(500);
104 }
105
106 /**************************************************************************/
107 /*
108     Display some basic info about the sensor status
109 */
110 /**************************************************************************/
111 void displaySensorStatus(void)
112 {
113   /* Get the system status values (mostly for debugging purposes) */
114   uint8_t system_status, self_test_results, system_error;
115   system_status = self_test_results = system_error = 0;
116   bno.getSystemStatus(&system_status, &self_test_results, &system_error);
117
118   /* Display the results in the Serial Monitor */
119   Serial.println("");
```

```
120   Serial.print("System Status: 0x");
121   Serial.println(system_status, HEX);
122   Serial.print("Self Test:     0x");
123   Serial.println(self_test_results, HEX);
124   Serial.print("System Error:  0x");
125   Serial.println(system_error, HEX);
126   Serial.println("");
127   delay(500);
128 }
129
130 /**************************************************************************/
131 /*
132     Display sensor calibration status
133 */
134 /**************************************************************************/
135 void displayCalStatus(void)
136 {
137   /* Get the four calibration values (0..3) */
138   /* Any sensor data reporting 0 should be ignored, */
139   /* 3 means 'fully calibrated" */
140   uint8_t system, gyro, accel, mag;
141   system = gyro = accel = mag = 0;
142   bno.getCalibration(&system, &gyro, &accel, &mag);
143
144   /* The data should be ignored until the system calibration is > 0 */
145   Serial.print("\t");
146   if (!system)
147   {
148     Serial.print("! ");
149   }
```

```
150
151   /* Display the individual values */
152   Serial.print("Sys:");
153   Serial.print(system, DEC);
154   Serial.print(" G:");
155   Serial.print(gyro, DEC);
156   Serial.print(" A:");
157   Serial.print(accel, DEC);
158   Serial.print(" M:");
159   Serial.print(mag, DEC);
160 }
161
162 /**************************************************************************/
163 /*
164     Callback for received data
165 */
166 /**************************************************************************/
167
168 void processMessage(int n) {
169
170   char ch = Wire.read();
171   if (ch == 'l') {
172     toggleLED();
173   }
174 }
175
176 /**************************************************************************/
177 /*
178     Method to toggle the LED. This is used for testing.
179 */
```

```
180 /*********************************************************************/
181
182 void toggleLED() {
183   ledOn = ! ledOn;
184   digitalWrite(ledPin, ledOn);
185
186 }
187
188 /*********************************************************************/
189 /*
190     Code that executes when request is received from Raspberry Pi
191 */
192 /*********************************************************************/
193
194 void sendIMUReading() {
195   Wire.write(imu_data, 12);
196 }
197
198 /*********************************************************************/
199 /*
200     Retrieves the digit of any position in an integer. The rightmost digit
201     has position 0. The second rightmost digit has position 1, etc.
202     e.g. Position 3 of integer 245984 is 5.
203 */
204 /*********************************************************************/
205
206 byte getDigit(int num, int n) {
207   int int_digit, temp1, temp2;
208   byte byte_digit;
209
```

```
210   temp1 = pow(10, n+1);
211   int_digit = num % temp1;
212
213   if (n > 0) {
214     temp2 = pow(10, n);
215     int_digit = int_digit / temp2;
216   }
217
218   byte_digit = (byte) int_digit;
219
220   return byte_digit;
221 }
222
223
224 /**************************************************************************/
225 /*
226     Arduino setup function (automatically called at startup)
227 */
228 /**************************************************************************/
229 void setup(void)
230 {
231     Serial.begin(9600);
232   while (!Serial)
233     ;
234
235   Serial.println("Orientation Sensor Starting...");
236   if (!bno.begin()) {
237     Serial.print("No BNO055 detected !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
238     while (1)
239       ;
```

```
240   }
241
242   delay(1000);
243
244   /* Display some basic information on this sensor */
245   displaySensorDetails();
246
247   /* Optional: Display current status */
248   displaySensorStatus();
249
250   bno.setExtCrystalUse(true);
251
252   pinMode(ledPin, OUTPUT); // This is used for testing.
253
254   Wire.begin(SLAVE_ADDRESS); // Set up the Wire library and make Arduino the slave
255
256   /* Define the callbacks for i2c communication */
257   Wire.onReceive(processMessage); // Used to specify a function when data received from Master
258   Wire.onRequest(sendIMUReading); // Used to specify a function when the Master requests data
259
260 }
261
262 /**************************************************************************/
263 /*
264     Arduino loop function, called once 'setup' is complete
265 */
266 /**************************************************************************/
267 void loop(void)
268 {
269
```

```
270  while (!done) {
271    /* Get a new sensor event */
272    sensors_event_t event;
273    bno.getEvent(&event);
274
275    /* Display the floating point data */
276    Serial.print("Yaw: ");
277    yaw = (int) event.orientation.x;
278    Serial.print(yaw);
279    if (yaw < 0) {
280      imu_data[8] = 1;  // Capture the sign information
281      yaw = abs(yaw);
282    }
283    else {
284      imu_data[8] = 0;
285    }
286    if (yaw > 360) {
287      yaw = yaw - 360; // Calculate equivalent angle
288    }
289
290    Serial.print("\tPitch: ");
291    pitch = (int) event.orientation.y;
292    Serial.print(pitch);
293    if (pitch < 0) {
294      imu_data[4] = 1;   // Capture the sign information
295      pitch = abs(pitch);
296    }
297    else {
298      imu_data[4] = 0;
299    }
```

```
300
301     Serial.print("\tRoll: ");
302     roll = (int) event.orientation.z;
303     Serial.print(roll);
304     if (roll < 0) {
305       imu_data[0] = 1;    // Capture the sign information
306       roll = abs(roll);
307     }
308     else {
309       imu_data[0] = 0;
310     }
311
312     /* Optional: Display calibration status */
313     displayCalStatus();
314
315     /* Optional: Display sensor status (debug only) */
316     //displaySensorStatus();
317
318     /* New line for the next sample */
319     Serial.println("");
320
321     /* Update the IMU data by extracting each digit from the raw data */
322     imu_data[1] = getDigit(roll, 2);
323     imu_data[2] = getDigit(roll, 1);
324     imu_data[3] = getDigit(roll, 0);
325     imu_data[5] = getDigit(pitch, 2);
326     imu_data[6] = getDigit(pitch, 1);
327     imu_data[7] = getDigit(pitch, 0);
328     imu_data[9] = getDigit(yaw, 2);
329     imu_data[10] = getDigit(yaw, 1);
```

```
330     imu_data[11] = getDigit(yaw, 0);
331
332     /* Wait the specified delay before requesting nex data */
333     delay(BNO055_SAMPLERATE_DELAY_MS);
334
335     end_program();
336
337   }
338
339   // Do nothing
340   while (true){};
341
342 }
```

On the Pi we can run a script to send IMU data over I2C and report IMU and GPS data to the console, but it's pretty boring, and we have to run another process like picam to show the camera feed in a browser, but this works to show the Roll+Pitch+Yaw and Lat+Lon+Alt:

```
1  from smbus2 import SMBus
2  from gps import *
3  import time
4  import threading
5
6  # for RPI version 1, use bus = SMBus(0)
7  bus = SMBus(1)
8
9  # This is the address we setup in the Arduino Program
10 SLAVE_ADDRESS = 0x04
11
12 gpsd = None
```

```
13
14 # Create a GPS Poller class
15 class GpsPoller(threading.Thread):
16   def __init__(self):
17     threading.Thread.__init__(self)
18     global gpsd
19     gpsd = gps(mode=WATCH_ENABLE)
20     self.current_value = None
21     self.running = True
22
23   def run(self):
24     global gpsd
25     while gpsp.running:
26       gpsd.next()
27
28 def request_reading():
29   # Read a block of 12 bytes starting at SLAVE_ADDRESS, offset 0
30   reading = bus.read_i2c_block_data(SLAVE_ADDRESS, 0, 12)
31
32   # Extract the IMU reading data
33   if reading[0] < 1:
34     roll_sign = "+"
35   else:
36     roll_sign = "-"
37   roll_1 = reading[1]
38   roll_2 = reading[2]
39   roll_3 = reading[3]
40
41   if reading[4] < 1:
42     pitch_sign = "+"
```

```
43    else:
44      pitch_sign = "-"
45    pitch_1 = reading[5]
46    pitch_2 = reading[6]
47    pitch_3 = reading[7]
48
49    if reading[8] < 1:
50      yaw_sign = "+"
51    else:
52      yaw_sign = "-"
53    yaw_1 = reading[9]
54    yaw_2 = reading[10]
55    yaw_3 = reading[11]
56
57    # Print the IMU and GPS data to the console
58    print("Roll: " + roll_sign + str(roll_1) + str(roll_2) + str(roll_3) +
59          "   Pitch: " + pitch_sign + str(pitch_1) + str(pitch_2) + str(pitch_3) +
60          "   Yaw: " + yaw_sign + str(yaw_1) + str(yaw_2) + str(yaw_3) +
61          "   Lat: " + str(gpsd.fix.latitude) +
62          "   Lon: " + str(gpsd.fix.longitude) +
63          "   Alt: " + str(gpsd.fix.altitude / .3048))
64
65  if __name__ == '__main__':
66    gpsp = GpsPoller()
67    try:
68      gpsp.start()
69      while True:
70        request_reading()
71        time.sleep(1)
72    except(KeyboardInterrupt, SystemExit):
```

```
    gpsp.running = False
    gpsp.join()
    bus.close()
```

Or we can make it a prettier webpage and include the camera feed and a HUD with the data we are trying to gather, using flask and opencv:

```
from smbus2 import SMBus
from gps import *
import time
import threading
from flask import Flask, render_template, Response, jsonify
import cv2

app = Flask(__name__)

# for RPI version 1, use bus = SMBus(0)
bus = SMBus(1)

# This is the address we setup in the Arduino Program
SLAVE_ADDRESS = 0x04

gpsd = None

# Create a GPS Poller class
class GpsPoller(threading.Thread):
  def __init__(self):
    threading.Thread.__init__(self)
    global gpsd
```

```
23      gpsd = gps(mode=WATCH_ENABLE)
24      self.current_value = None
25      self.running = True
26
27    def run(self):
28      global gpsd
29      while gpsp.running:
30        gpsd.next()
31
32  def request_reading():
33    # Read a block of 12 bytes starting at SLAVE_ADDRESS, offset 0
34    reading = bus.read_i2c_block_data(SLAVE_ADDRESS, 0, 12)
35
36    # Extract the IMU reading data
37    if reading[0] < 1:
38      roll_sign = "+"
39    else:
40      roll_sign = "-"
41    roll_1 = reading[1]
42    roll_2 = reading[2]
43    roll_3 = reading[3]
44
45    if reading[4] < 1:
46      pitch_sign = "+"
47    else:
48      pitch_sign = "-"
49    pitch_1 = reading[5]
50    pitch_2 = reading[6]
51    pitch_3 = reading[7]
52
```

```
53  if reading[8] < 1:
54    yaw_sign = "+"
55  else:
56    yaw_sign = "-"
57  yaw_1 = reading[9]
58  yaw_2 = reading[10]
59  yaw_3 = reading[11]
60
61  # Create a dictionary with the IMU and GPS data
62  data = {
63    'roll': roll_sign + str(roll_1) + str(roll_2) + str(roll_3),
64    'pitch': pitch_sign + str(pitch_1) + str(pitch_2) + str(pitch_3),
65    'yaw': yaw_sign + str(yaw_1) + str(yaw_2) + str(yaw_3),
66    'latitude': str(gpsd.fix.latitude),
67    'longitude': str(gpsd.fix.longitude),
68    'altitude': str(gpsd.fix.altitude / .3048)
69  }
70
71  return data
72
73 def gen_frames():
74    camera = cv2.VideoCapture(0)  # Use 0 for the first connected USB camera
75    while True:
76        success, frame = camera.read()
77        if not success:
78            break
79        else:
80            ret, buffer = cv2.imencode('.jpg', frame)
81            frame = buffer.tobytes()
82            yield (b'--frame\r\n'
```

```
83                 b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
84
85 @app.route('/video_feed')
86 def video_feed():
87     return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
88
89 @app.route('/')
90 def index():
91     return render_template('index.html')
92
93 @app.route('/data')
94 def data():
95     # Read the IMU and GPS data
96     imu_gps_data = request_reading()
97
98     # Return the data as JSON
99     return jsonify(imu_gps_data)
100
101 if __name__ == '__main__':
102     gpsp = GpsPoller()
103     try:
104         gpsp.start()
105         app.run(host='0.0.0.0', port=5000)
106     except(KeyboardInterrupt, SystemExit):
107         gpsp.running = False
108         gpsp.join()
109         bus.close()
```

This is a much nicer view of the data and gives us a first person perspective on the drone, you might even be able to fly it like this!

Demo

The data is successfully captured, and Roll+Pitch+Yaw, Lat+Lon+Alt, and a camera feed are viewable over a Wifi connection on a remote machine, the drone flies and the data is sent over in flight.

https://drive.google.com/file/d/1_j6ItqwzVEt2p98rxCX369myHtrReaNc/view?usp=sharing

References

https://automaticaddison.com/how-to-send-roll-pitch-yaw-data-over-i2c-from-arduino-to-raspberry-pi/