Code Development

Blue Coding Team:
Kevin Parlak
Timothy Sam
Nawaf Abdullah
Evan Kerr
Ali Wahab

AERSP 440; Blue Team Gantt Chart

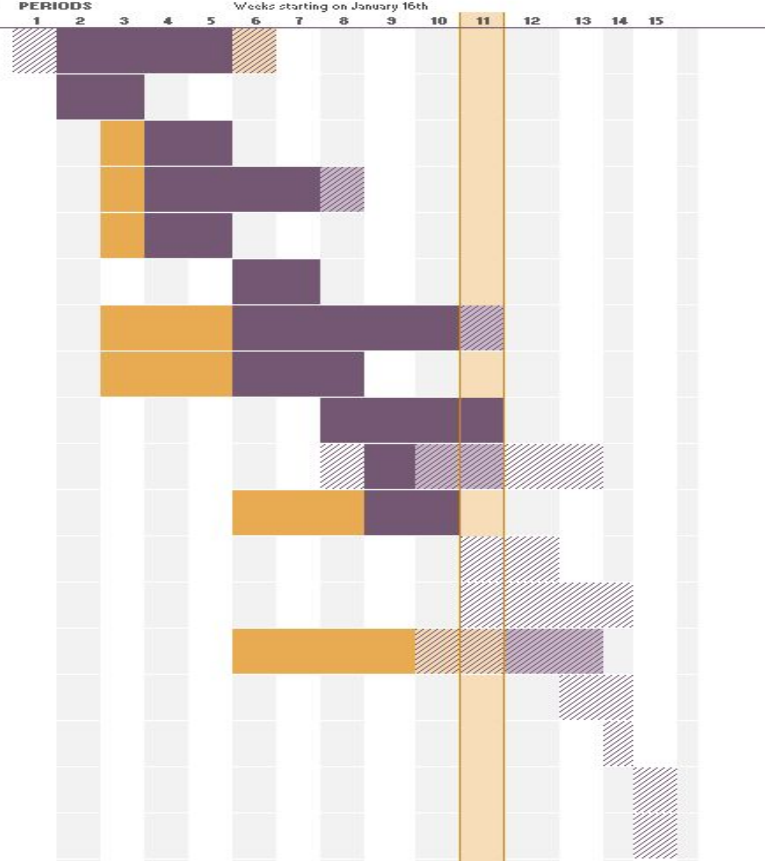
Select a period to highlight at right. A legend describing the charting follows.

Period Highlig 11

% Complete (beyond plan)

Plan Duration | Actual Start | % Complete | Actual (beyond plan)

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Requirements	1	5	2	5	95%
"What is Requirements"	2	2	2	2	100%
Requirements Presentation	4	2	3	3	100%
Design	4	5	3	6	95%
"What is Design"	4	2	3	3	100%
Design Presentation	6	2	6	2	100%
Coding	6	6	3	9	90%
"What is good Coding"	6	3	3	6	100%
Coding Presentation	8	4	8	4	100%
Testing	8	6	9	3	10%
"What is Testing"	9	2	6	5	100%
Testing Presentation	11	2	0	0	0%
V&V	11	4	0	0	0%
"What is V&V"	12	2	6	8	60%
V&V Presentation	13	2	0	0	0%
Dry Run	14	1	0	0	0%
Final Competition	15	1	0	0	0%
Submit all documentation	15	1	0	0	0%

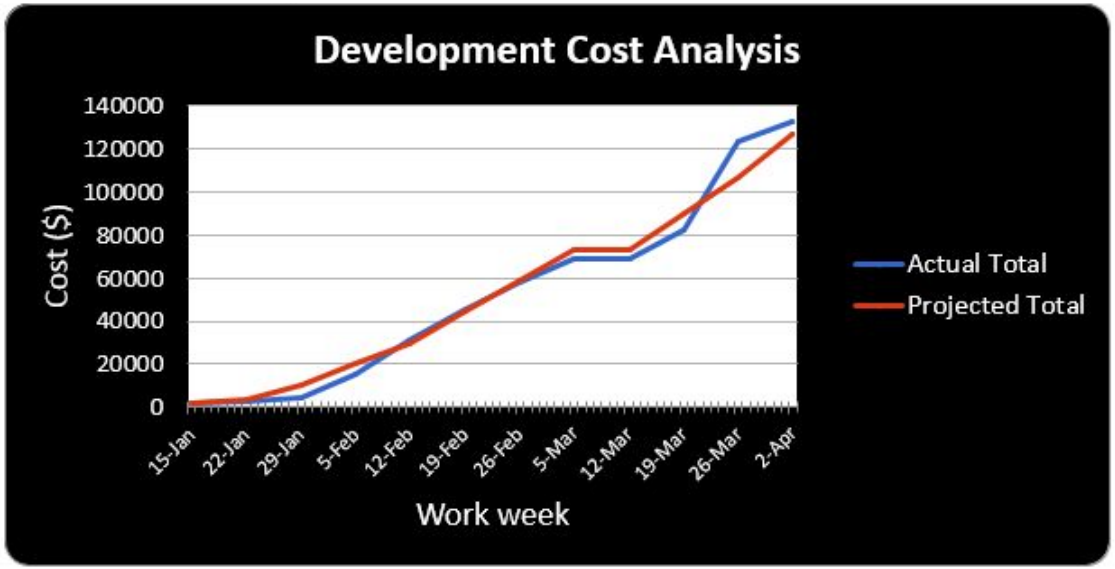


Financial Progress Report

COCOMO Estimation	
Type	Cost ($)
Organic	172934.40
Semi-detached	216500.40
Embedded	269219.27

Final estimated bill: $191200

Total current costs: $127200

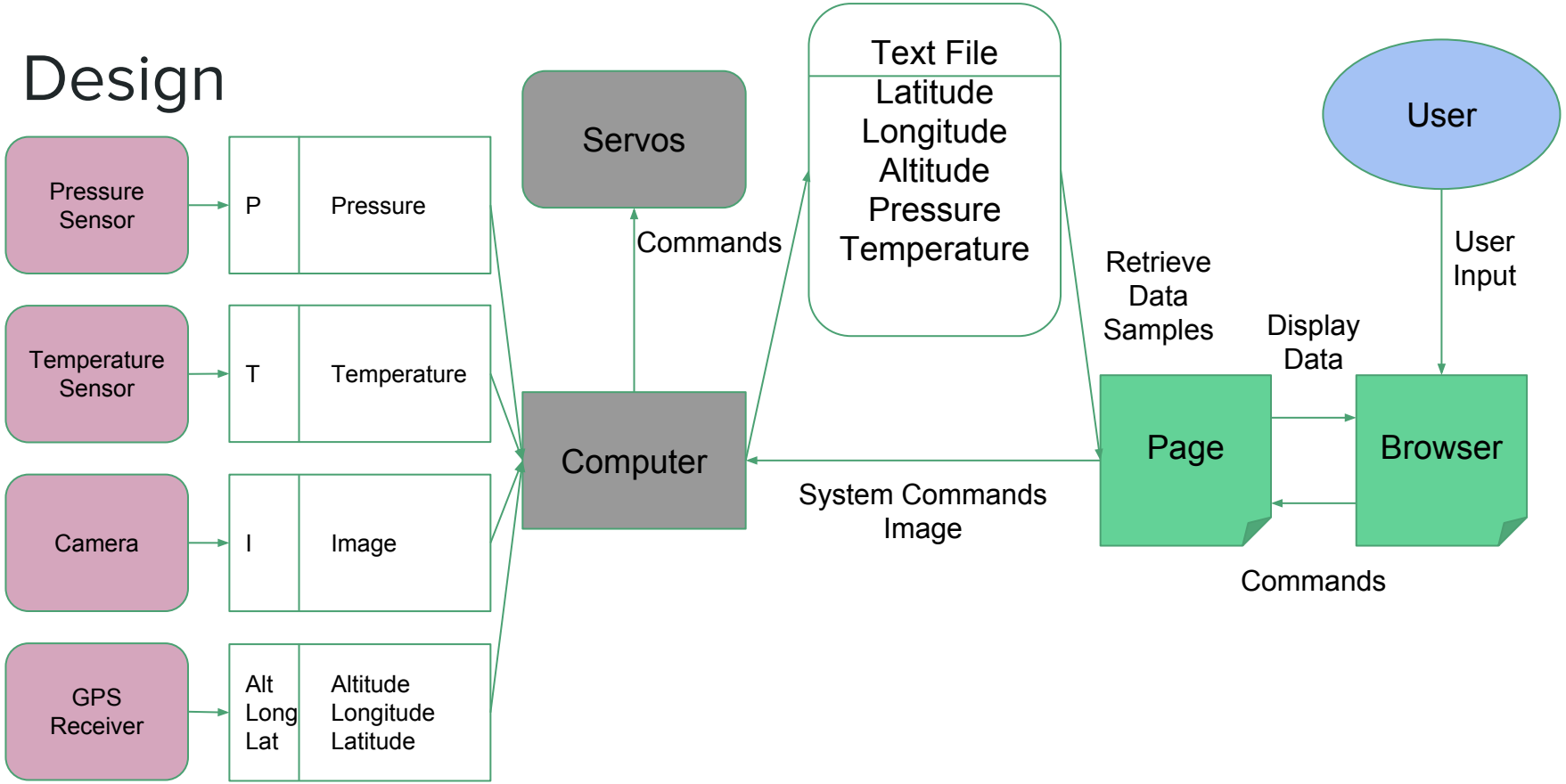


Requirements

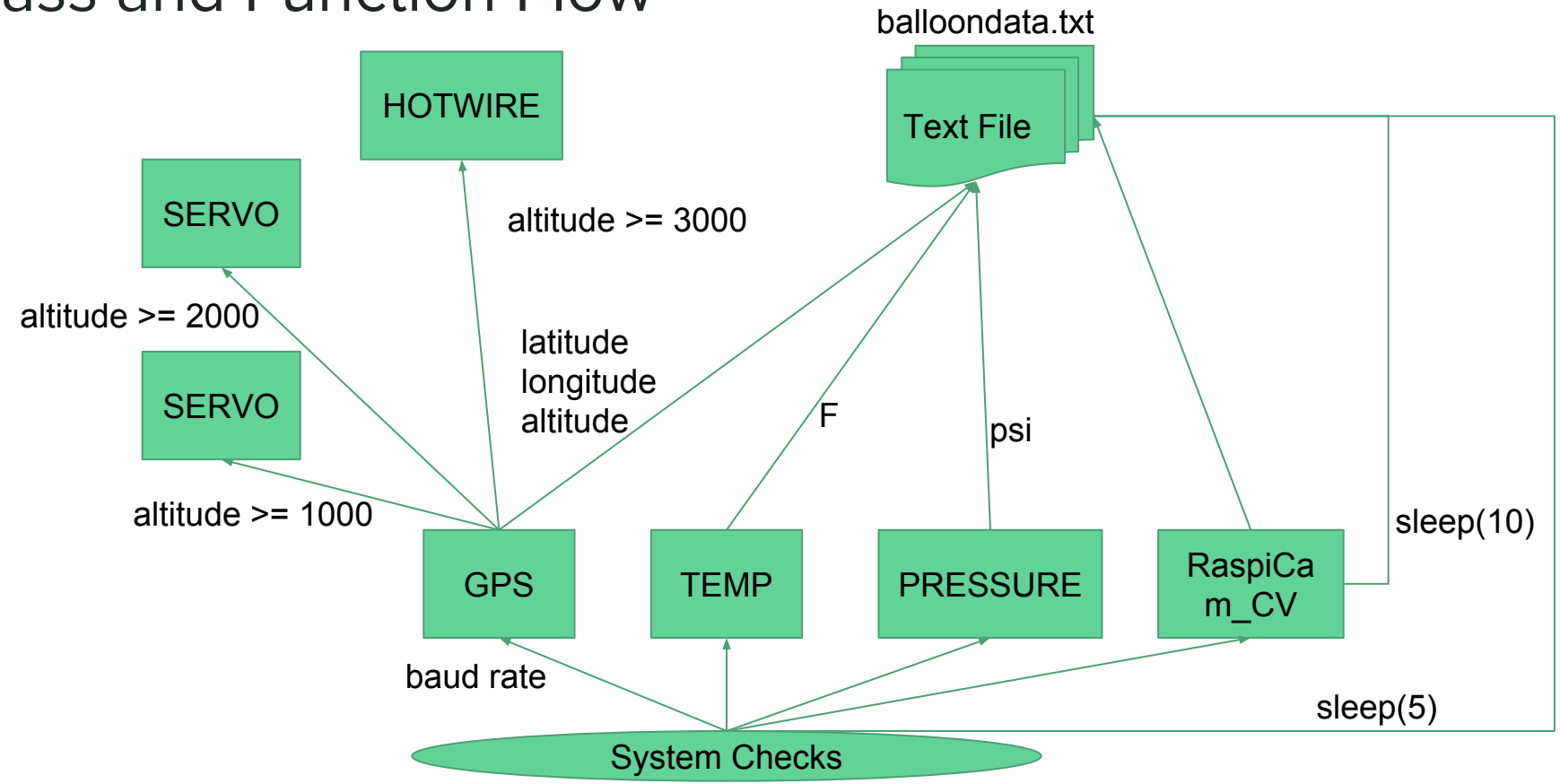
System Level Requirements

- User Interface
  - **Webhost** - The user interface shall display flight system instrument readings on a laptop within a single window that shows live measurements and graphs data as a function of altitude.
- Flight Payload
  - **Database** - The flight computer shall continuously measure data from onboard instruments.
  - **Webhost** - The flight computer shall transmit data from onboard instruments to the user interface.
  - **GPS, SERVO** - The flight payload shall autonomously unfurl a red ribbon when altitude data reaches 1000 ft.
  - **GPS, SERVO** - The flight payload shall autonomously unfurl a black/white ribbon when altitude data reaches 2000 ft.
  - **GPS, HOTWIRE** - The flight payload shall autonomously release the payload when altitude data reaches 3000 ft.
  - **Executables** - The flight payload shall receive, process, and execute commands from the user interface as a redundant safety measure.
  - **HOTWIRE** - A parachute shall autonomously deploy following payload separation.

Design


Pressure Sensor
P
Pressure
Temperature Sensor
T
Temperature
Camera
I
Image
GPS Receiver
Alt
Long
Lat
Altitude
Longitude
Latitude
Servos
Commands
Computer
Text File
Latitude
Longitude
Altitude
Pressure
Temperature
Retrieve Data Samples
System Commands
Image
User
User Input
Display Data
Page
Browser
Commands

Class and Function Flow



GPS Unit

Use of NMEA Parser algorithm:

- GPGGA data only
  - Latitude, longitude, and altitude
- Parsed by separating data into vectors by comma separation
  - splitStringByComma
- Two functions determine the parse
  - isValidGGA
  - setValuesGGA

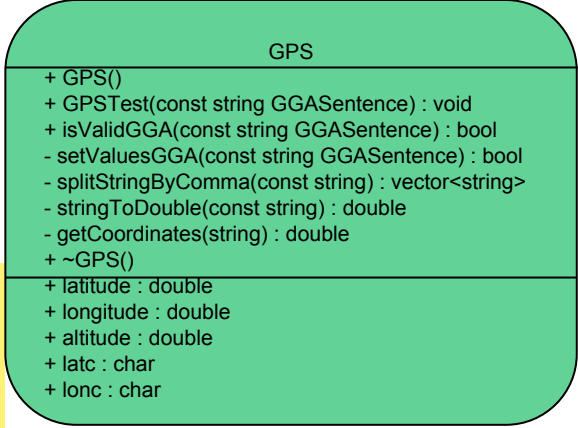
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47

OOP Advantage:

- Parsing scheme hidden in object

GPS Unit (cont.)

```
//**********Latitude, Longitude, Altitude Data**********
        //Set data to bad to start
        bool goodGPS=false;
        //Loop until a GPGGA point is read with good data
        while(goodGPS!=true)
        {
            //Stream data from GPS device
            f>>nmea;
            //Call GPS class
            gps.GPSTest(nmea);
                //Is the data GPGGA data only?
                if(gps.isValidGGA(nmea))
                {
                    //Save latitude data to overwritten file
                    balloondata<<setprecision(6)<<gps.latitude<<" "<<gps.latc<<endl;
                    //Save latitude data to appended file
                    balloondataSAVED<<setprecision(6)<<gps.latitude<<" "<<gps.latc<<endl;
                    //Save longitude data to overwritten file
                    balloondata<<setprecision(6)<<gps.longitude<<" "<<gps.lonc<<endl;
                    //Save longitude data to appended file
                    balloondataSAVED<<setprecision(6)<<gps.longitude<<" "<<gps.lonc<<endl;
                    //Save altitude data to overwritten file
                    balloondata<<setprecision(5)<<gps.altitude<<" ft"<<endl;
                    //Save altitude data to appended file
                    balloondataSAVED<<setprecision(5)<<gps.altitude<<" ft"<<endl;
                    //Good data has been recorded, exit GPS
                    goodGPS=true;
                }
        }//End while loop - GPS
```



Camera Unit

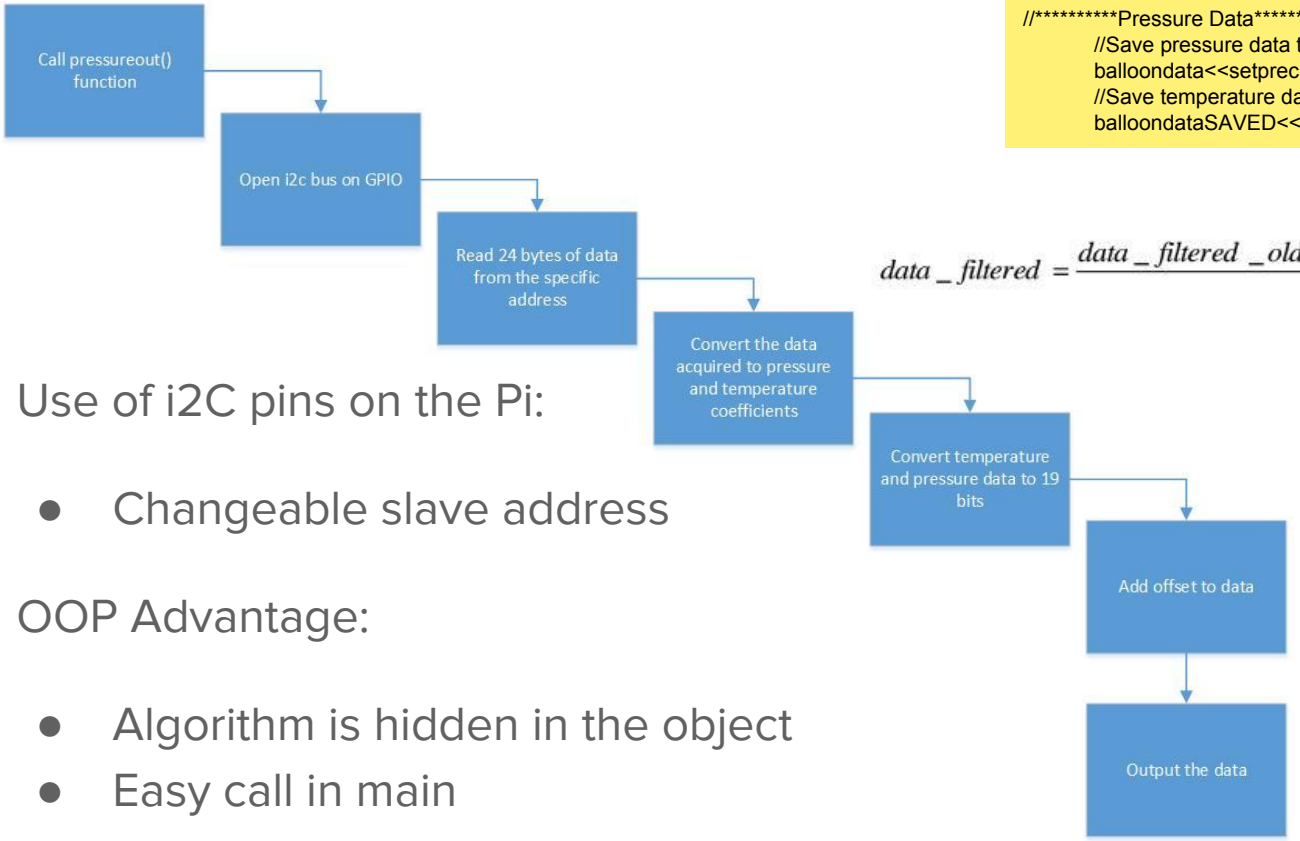
Use of open source Raspicam libraries:

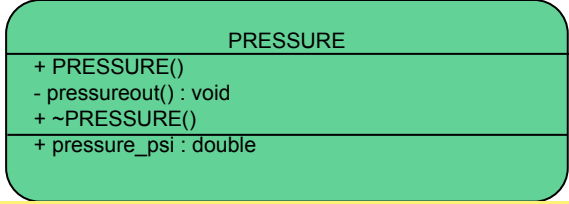
- Holds all functions and serial calls to Raspberry Pi Camera port
- Open_CV used in conjunction with RaspiCam
- https://github.com/cedricve/raspicam.git
- https://github.com/opencv/opencv.git

```
//**********Camera Data**********
        //Counter to take pictures every 10 seconds, uses divisor check
        if(count%2==0)
        {
            //Folder location
            string path = "/media/pi/98E1-3336/Data/";
            //Type of picture
            string picfile = ".jpg";
            //Overall filename for image
            string filename;
            //Incrementing integer for saved images
            string fileint;
            //Link variables
            filename.append(path);
            //Convert integer to string and append
            ostringstream convert;
            convert<<piccnt;
            fileint=convert.str();
            filename=filename+fileint;
            filename.append(picfile);
            //Convert filename to C string for code
            const char *file=filename.c_str();

            //Start capture
            for(int i=0;i<nCount;i++)
            {
               Camera.grab();
               Camera.retrieve(image);
            }
            //SAVED file name
            cv::imwrite(file,image);
            //OVERWRITTEN file name
            cv::imwrite("image.jpg",image);
            //Increment picture count
            piccnt++;
        }
        //Wait 10 seconds for new data
        count++;
```

Pressure Sensor






```
//**********Pressure Data**********
        //Save pressure data to overwritten file
        balloondata<<setprecision(4)<<pressure.pressure_psi<<" psi"<<endl;
        //Save temperature data to appended file
        balloondataSAVED<<setprecision(4)<<pressure.pressure_psi<<" psi"<<endl;
```

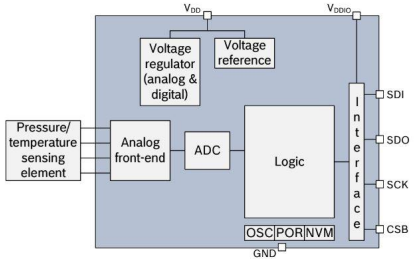
$$data\_filtered = \frac{data\_filtered\_old \cdot (filter\_coefficient - 1) + data\_ADC}{filter\_coefficient},$$

Use of i2C pins on the Pi:

- Changeable slave address

OOP Advantage:

- Algorithm is hidden in the object
- Easy call in main



Temperature Sensor

TEMP
+ TEMP() - tempout() : void + ~TEMP()
+ temp_F : double

```
//**********Temperature Data**********
        //Save temperature data to overwritten file
        balloondata<<setprecision(3)<<temp.temp_F<<" F"<<endl;
        //Save temperature data to appended file
        balloondataSAVED<<setprecision(3)<<temp.temp_F<<" F"<<endl;
```

Option 1 - DHT-11 Sensor

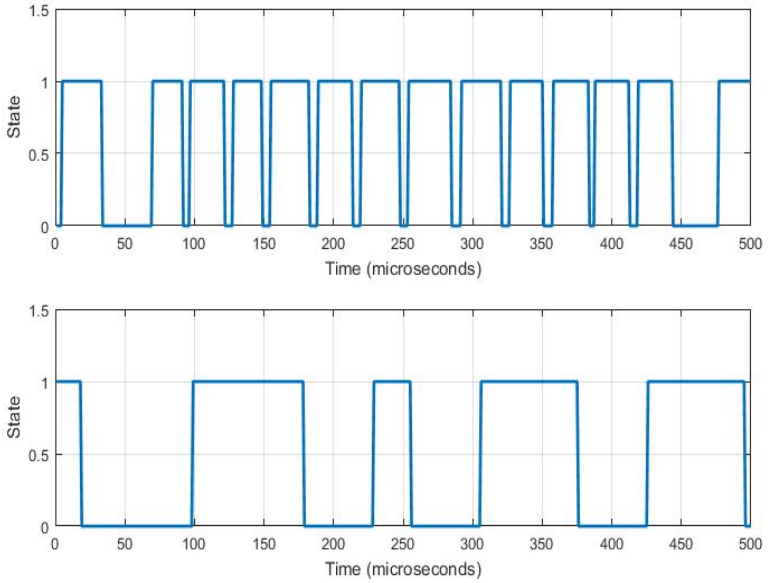
- Noisy data, high error rate
- Complicated read process

Option 2 - BMP280 Pressure Sensor

- Includes high quality temp sensor

OOP Advantage:

- Algorithm is hidden in the object
- Easy call in main



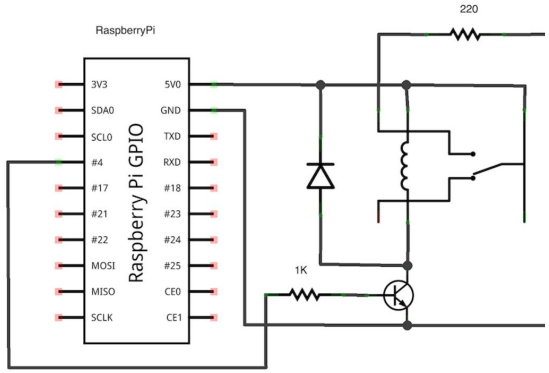
Hot Wire

HOTWIRE
+ HOTWIRE() + HOTWIREFIRE(int time) : void + ~HOTWIRE()

```
//**********Hotwire**********
        //Ignite nichrome wire and release mechanism at AGL >= 3000 ft
          if(gps.altitude>=2900)
          {
            //Fire hotwire circuit for specified amount of time
            hotwire.HOTWIREFIRE(20);
            //Shut off data streaming
            hotwireON=true;
            //Error output check
            if(hotwireON==true)
            {
              cout<<"Hotwire is ON..."<<endl;
            }
          }
```

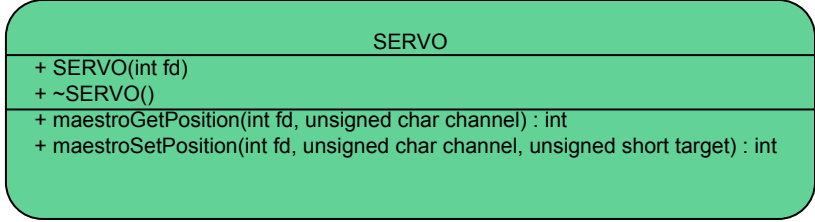
Use of open source WiringPi libraries:

- GPIO output signal
- https://github.com/WiringPi/WiringPi.git
- GPIO pin to send 3.3V signal to transducer
- Transducer to allow voltage to flow from external 9V battery through relay
- Relay to control 9V battery flow through nichr



OOP Advantage:

- Easy call in main
- Simple, clear function



Servo Motors

Use of open source Pololu code:

- Call to USB port microcontroller is connected to
- https://www.pololu.com/docs/0J40/5.h.1

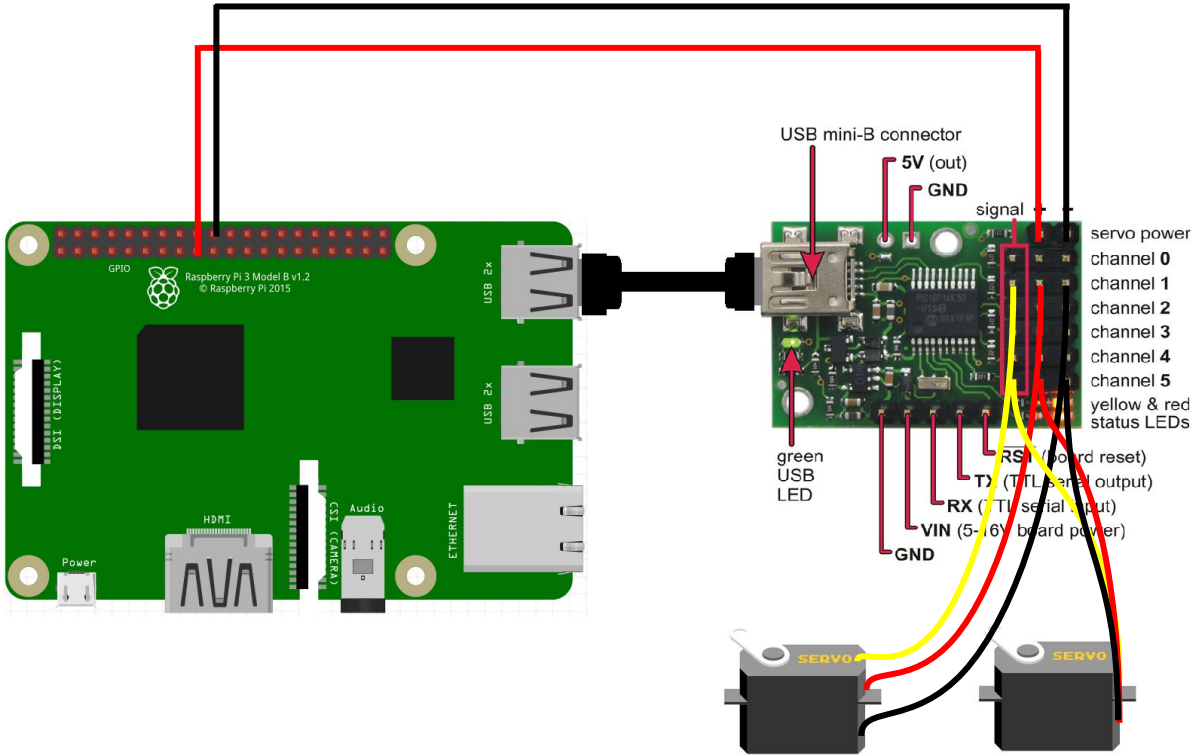
OOP Advantage:

- Switchable channel call in main
- Position setting in main
- Ability to call more than one servo

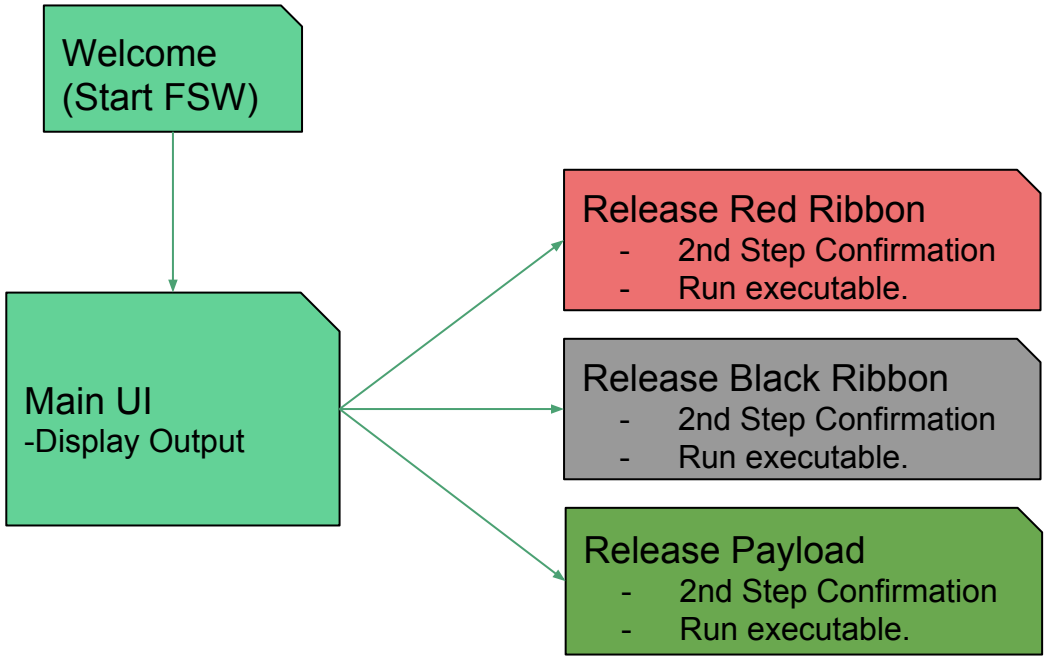
Servo Motors (cont.)

```
//**********Servos**********
        //Altitude = 1000 feet AGL?
        if(gps.altitude>=900&&servo1000DONE!=true)
        {
           //Call servo class with channel from Maestro
           //Set max position for servo1000
           servo1000.maestroSetTarget(fd,1,30000);
           //Reset
           servo1000DONE=true;
        }
           //Reset servo1000
           if(servo1000DONE==true&&servoZERO!=true)
           {
              sleep(1);
              //Call servo class with channel 1 from Maestro
              //Reset position to 0
              servo1000.maestroSetTarget(fd,1,0);
              servoZERO=true;
           }
        //Altitude = 2000 feet AGL?
        if(gps.altitude>=1900&&servo2000DONE!=true)
        {
           //Call servo class with channel from Maestro
           //Set max position for servo2000
           servo2000.maestroSetTarget(fd,5,30000);
           //Reset
           servo2000DONE=true;
        }
           //Reset servo2000
           if(servo2000DONE==true)
           {
              sleep(1);
              //Call servo class with channel 5 from Maestro
              //Reset position to 0
              servo2000.maestroSetTarget(fd,5,0);
              servo2ZERO=true;
           }
        //Close port
        close(fd);
```

Servo - Pololu - Pi Wire Schematic

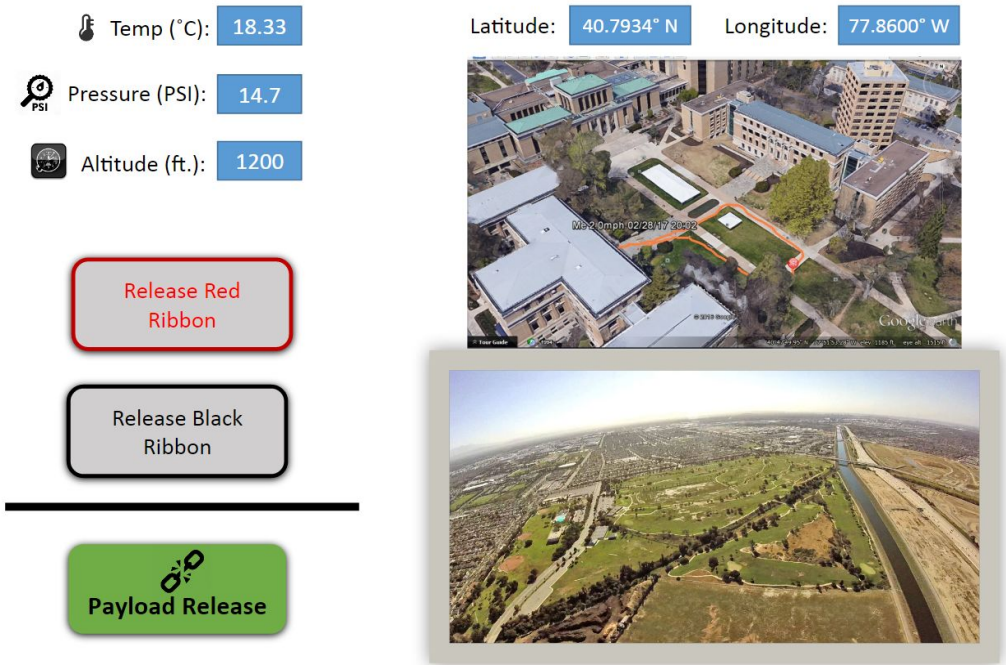


User Interface



- Implemented utilizing Apache Web Server Software
- Consists of HTML pages and CSS style sheets
- Tasks executed using PHP scripting language

User Interface Before Development

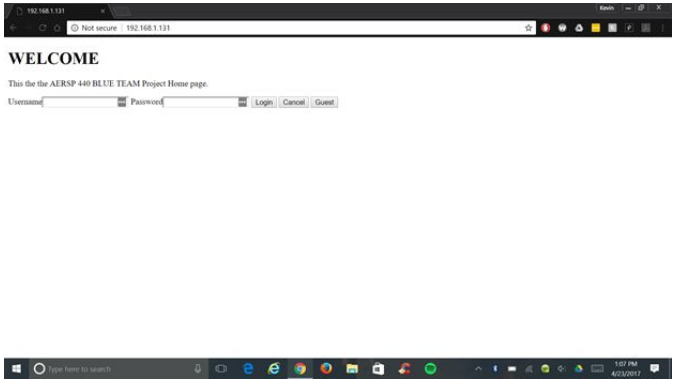


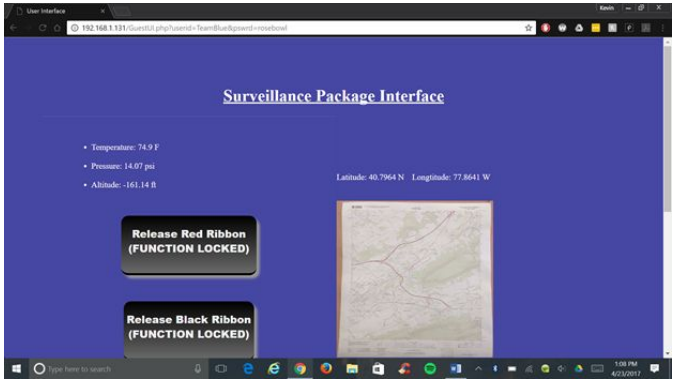
User Interface After Development (Tentative)

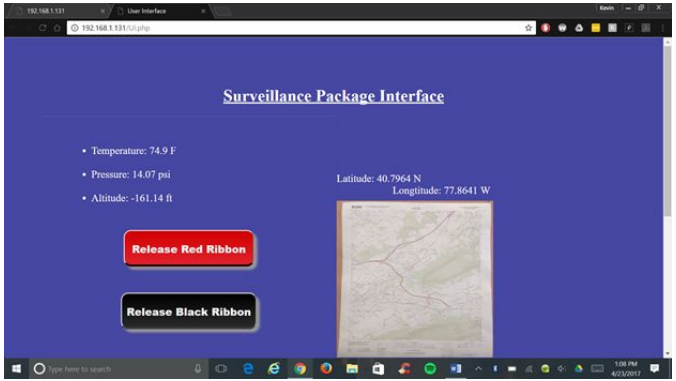
Features:

- Written as a webpage in HTML & CSS
- Displays temperature, pressure, altitude, latitude, and longitude.
- Three buttons for redundancy that are linked to executables.
- Displays a map with the package's location.
- Displays images captured with RasPi Camera.
- Refreshes automatically every 5 seconds.

Final User Interface







Data

UI

- Temperature (°C):
- Pressure (psi):
- Altitude (ft):

Latitude (°N): Longtitude (°W):

HTML

```
30    <ul><font color="white">
31      <li>Temperature (&deg;C): <?php   echo $data[0]   ?> </li>
32      <br>
33      <li>Pressure (psi): <?php echo $data[1] ?></li> <!-- pressu
34      <br>
35      <li>Altitude (ft): <?php echo $data[2] ?></li> <!-- altitud
36        </font>
37    </ul>
```

CSS

```
1   .wrapper {
2       width:600px;
3       margin: 0 auto;
4   }
5
6   #latitude {
7       float:left;
8   }
9
10  #longtitude {
11      float:right;
12  }
```

Data

UI

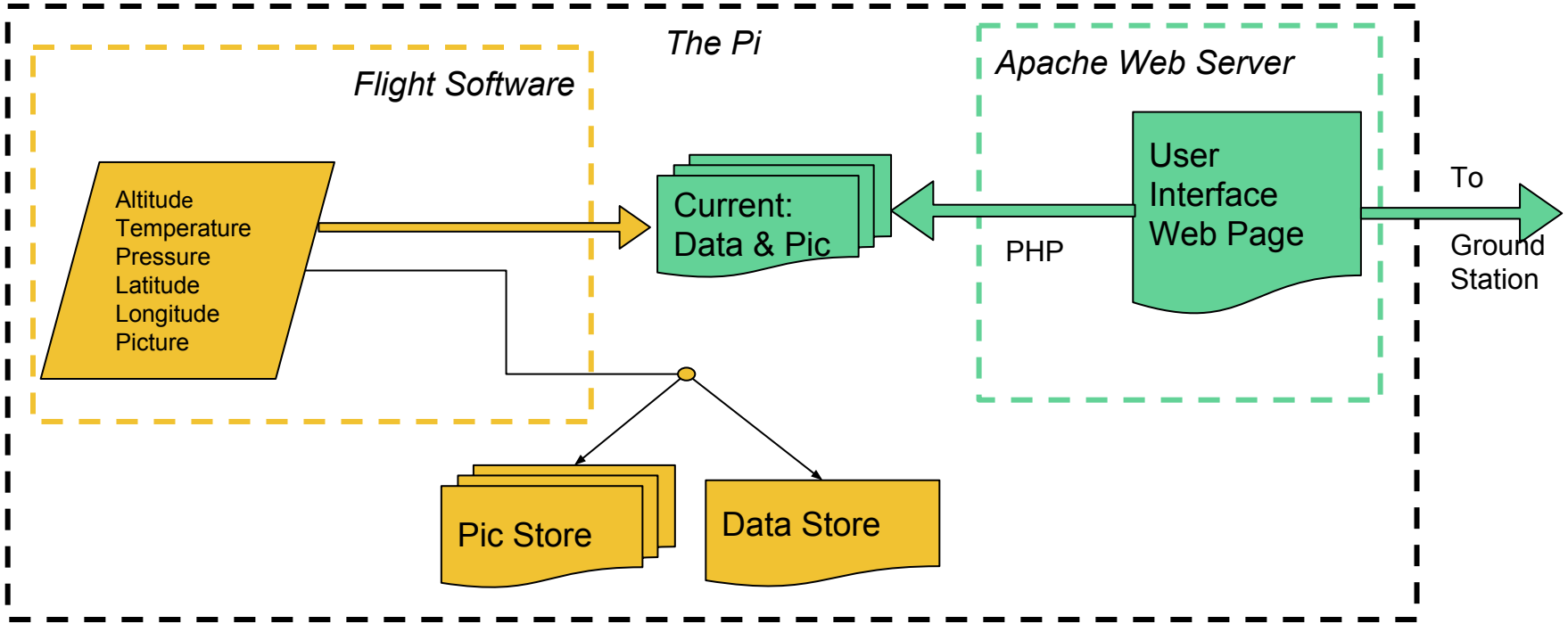
HTML

```
<!-- /// Ribbon buttons - start /// -->
<div>
    <form action="servo.php" method="post">
    <input type="hidden" name="color" value="red" />
    <input type="submit" id="button1" value="Release Red Ribbon" />
    </form>
```

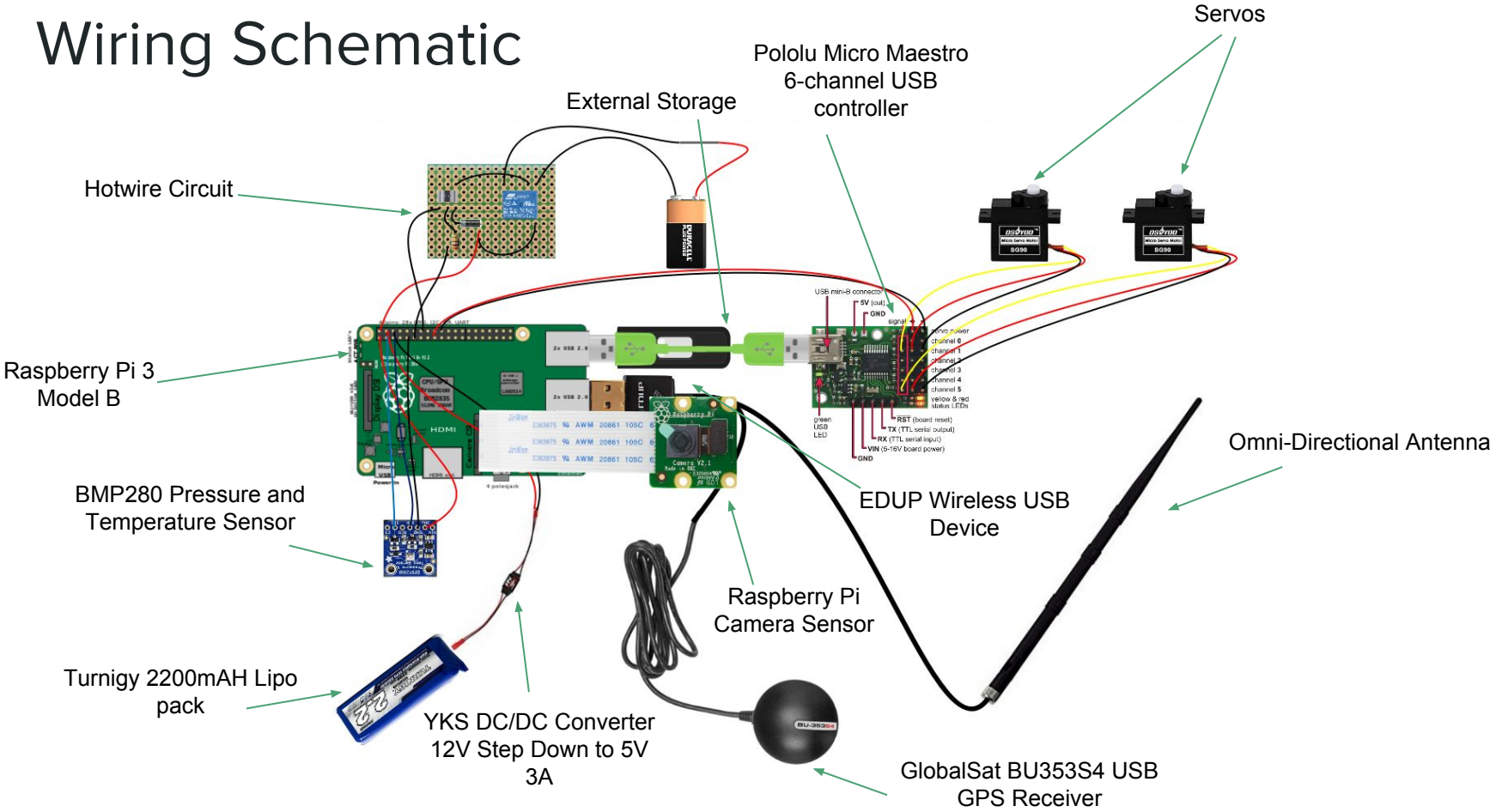
CSS

```
#button1:active {
    box-shadow: 2px 2px 2px #777;
    border-bottom:1px solid #230001;
    transform: translateY(3px);
}
```

Database


The Pi
Flight Software
Altitude
Temperature
Pressure
Latitude
Longitude
Picture
Current:
Data & Pic
Apache Web Server
User
Interface
Web Page
PHP
To
Ground
Station
Pic Store
Data Store

Wiring Schematic



Using the Software

Startup:

- Just need IP Address for the Pi
- Putty to Secure Shell into Pi to start flight software
  - http://www.putty.org/
  - See any command line outputs during System Check via terminal on Putty
- Welcome page
  - Login as Guest or User

Operation:

- Main UI
  - Display last current data values and image
  - Manually issue commands to release ribbons or payload

Testing

GPS

- Determine margin of error for altitude readings

Camera

- Begin Unit Tests on Camera Class

Sensors

- Begin Unit Tests on Pressure and Temperature Classes

Hotwire

- Hardware Tests
- Unit test code with hardware

Servos

- Unit test Servo Class to set servo position
- Implement with ribbon deployment system

UI

- Access UI from PC
- Create small executables to be called from button clicks (i.e. stubs)

Questions?

01010001011101010110010101110011011101000110100101101111011011100111001100111111