Hunter Cobb
KUID: 2913822

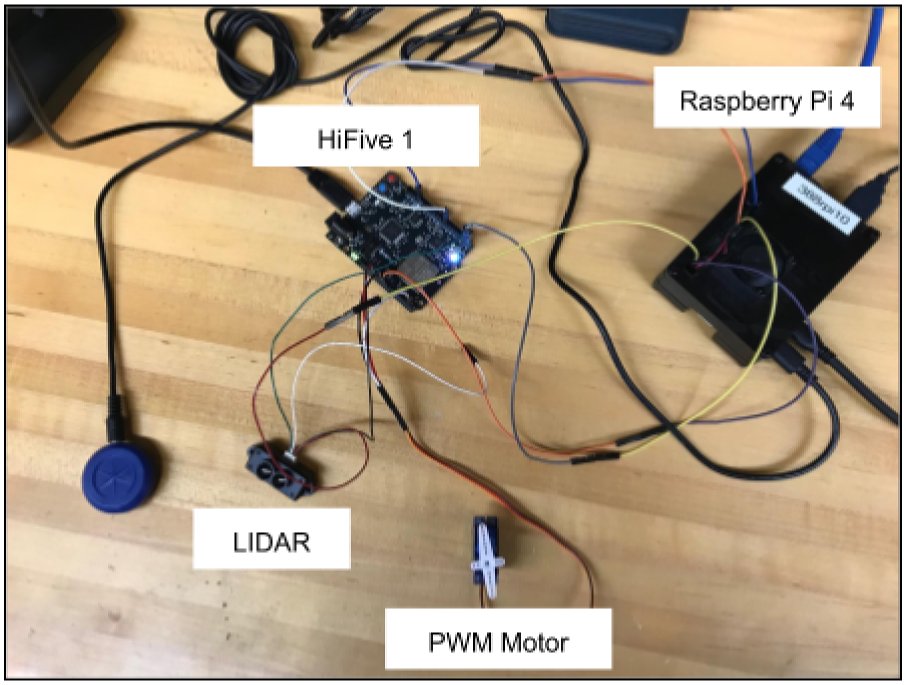
EECS 388 Final Project

Putting All Things Together: (Almost) Self-Driving Car

12/2/19
8:00 am

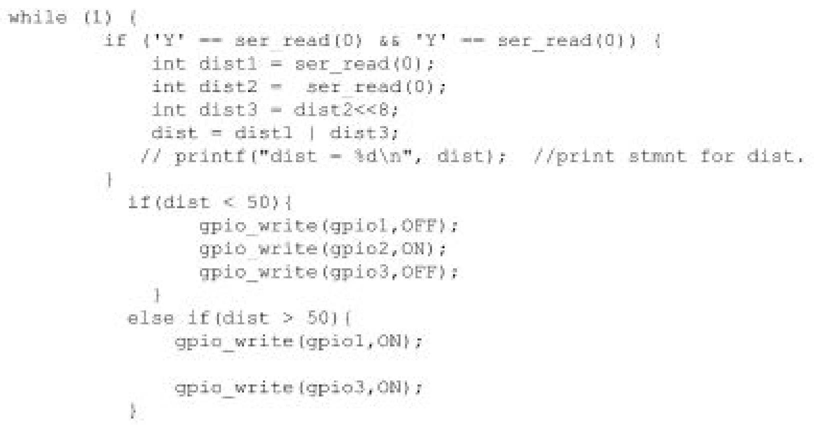
Procedure:

The final project that we have demonstrated knowledge of today is a culmination of the last semester's worth of labs and work into a single final project that simulates the behaviour of a self driving car using a Raspberry Pi 4, HiFive-1 board, TFMini LIDAR, and the mini pwm motor. The total hardware system is pictured below.



As we can see, there is quite a bit going on here on the hardware side of this system with multiple jumper wires running from the sensors, actuators, and the boards themselves.

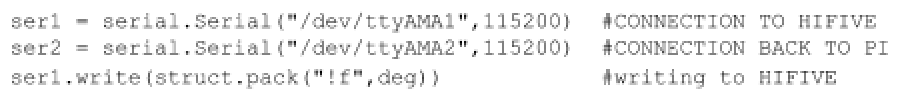
The first step of this project was to setup the LIDAR using C. The code itself is quite simple and uses polling I/O on the HiFive 1 to get the data from the LIDAR sensor and change the LED in accordance with the distance read by the LIDAR. When the LIDAR receives a value below 50 cm, the program writes the red LED to be the only light on to signify that distance < 50 cm and turns the PWM off to avoid a collision.The LED turns back to White When the LIDAR receives dist > 50. Below is an excerpt from “comm.c” where the LIDAR is implemented in the project.



```
while (1) {
        if ('Y' == ser_read(0) && 'Y' == ser_read(0)) {
            int dist1 = ser_read(0);
            int dist2 =  ser_read(0);
            int dist3 = dist2<<8;
            dist = dist1 | dist3;
           // printf("dist = %d\n", dist);  //print stmnt for dist.
        }
          if(dist < 50){
                gpio_write(gpio1,OFF);
                gpio_write(gpio2,ON);
                gpio_write(gpio3,OFF);
            }
          else if(dist > 50){
              gpio_write(gpio1,ON);

              gpio_write(gpio3,ON);
          }
```

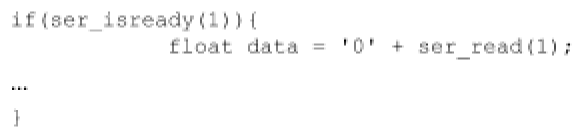
The next section of the project moves on to the actual reading of the input data given to the HiFive from the Raspberry Pi system. To accomplish this, we had to setup a connection between the HiFive 1 and the Raspberry Pi such that the projected angle from the Pi can be sent and used by the microcontroller to control the PWM motor to simulate the “car” turning. Below are two excerpts, one that shows the Raspberry Pi sending information and the other to show the HiFive receiving the data.

Python(dnn.py):



```
ser1 = serial.Serial("/dev/ttyAMA1",115200)  #CONNECTION TO HIFIVE
ser2 = serial.Serial("/dev/ttyAMA2",115200)  #CONNECTION BACK TO PI
ser1.write(struct.pack("!f",deg))            #writing to HIFIVE
```

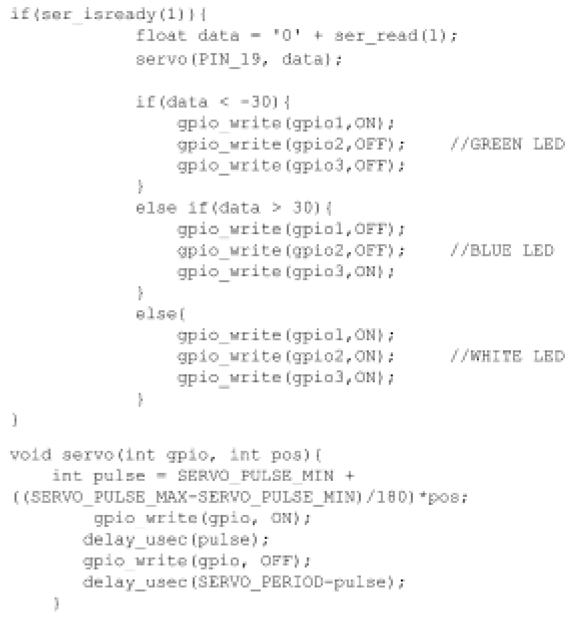
C(comm.c):



```
if(ser_isready(1)){
            float data = '0' + ser_read(1);
...
}
```

The python is run on the Raspberry and establishes a serial connection between the two boards and later writes to the HiFive connection which the HiFive Receives and grabs with the C code shown and is used for the PWM motor and LEDs which is the next segment of this project.

The projects main functionality is to control LEDs and a PWM motor based on data passed between the two boards as shown above. The actual control of the PWM and LEDs is shown in the code below.

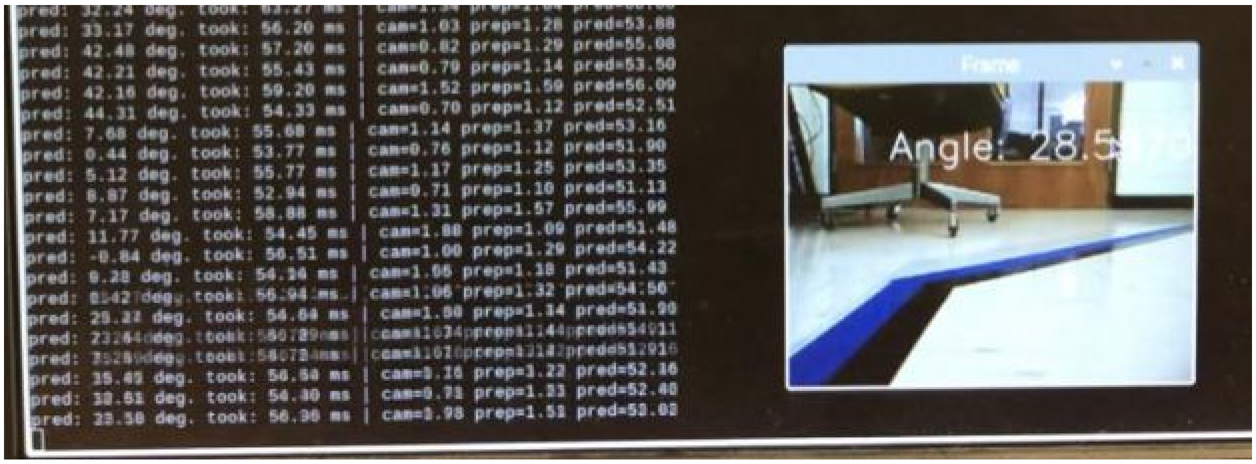


```
if(ser_isready(1)){
            float data = '0' + ser_read(1);
            servo(PIN_19, data);

            if(data < -30){
                gpio_write(gpio1,ON);
                gpio_write(gpio2,OFF);     //GREEN LED
                gpio_write(gpio3,OFF);
            }
            else if(data > 30){
                gpio_write(gpio1,OFF);
                gpio_write(gpio2,OFF);     //BLUE LED
                gpio_write(gpio3,ON);
            }
            else{
                gpio_write(gpio1,ON);
                gpio_write(gpio2,ON);      //WHITE LED
                gpio_write(gpio3,ON);
            }
}

void servo(int gpio, int pos){
    int pulse = SERVO_PULSE_MIN +
((SERVO_PULSE_MAX-SERVO_PULSE_MIN)/180)*pos;
        gpio write(gpio, ON);
       delay_usec(pulse);
       gpio write(gpio, OFF);
       delay_usec(SERVO_PERIOD-pulse);
    }
```

The code above is from the HiFive Board and waits for the serial input from the Pi to be ready, and when ready, reads data from the board and calls the servo function which acts as a converter to allow the degree to be given to the servo in the correct format and move the motor the correct amount. The code after the “servo” function call simply changes the LED color based on the reading of the angle from the Pi. This concludes the requirements for part one of the project and the next section will be about part two.

The second section of the project dealt more with UI than hardware/software relations and is completely done on the Raspberry Pi. The goal of the second part of this project was to make a new window that displays the video as well as the current output angle of the car on the screen for the user. The following code snippet was used to achieve this result as well as an included photo of the screen itself during a run.

```
ret, frame = cap.read()
cv2.putText(frame,"Angle: " + str(deg), (80, 60), cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255, 255, 255), lineType=cv2.LINE_AA)
if ret == True:
      # Display the resulting frame
      cv2.imshow('Frame',frame)

      # Press Q on keyboard to  exit
      if cv2.waitKey(25) & 0xFF == ord('q'):
           break
```



```
pred: 32.24 deg. took: 63.27 ms | [illegible]
pred: 33.17 deg. took: 56.20 ms | cam=1.03 prep=1.28 pred=53.88
pred: 42.48 deg. took: 57.20 ms | cam=0.82 prep=1.29 pred=55.08
pred: 42.21 deg. took: 55.43 ms | cam=0.79 prep=1.14 pred=53.50
pred: 42.16 deg. took: 59.20 ms | cam=1.52 prep=1.59 pred=56.09
pred: 44.31 deg. took: 54.33 ms | cam=0.70 prep=1.12 pred=52.51
pred: 7.68 deg. took: 55.68 ms | cam=1.14 prep=1.37 pred=53.16
pred: 0.44 deg. took: 53.77 ms | cam=0.76 prep=1.12 pred=51.90
pred: 5.12 deg. took: 55.77 ms | cam=1.17 prep=1.25 pred=53.35
pred: 8.87 deg. took: 52.94 ms | cam=0.71 prep=1.10 pred=51.13
pred: 7.17 deg. took: 58.88 ms | cam=1.31 prep=1.57 pred=55.99
pred: 11.77 deg. took: 54.45 ms | cam=1.88 prep=1.09 pred=51.48
pred: -0.84 deg. took: 56.51 ms | cam=1.00 prep=1.29 pred=54.22
pred: 0.28 deg. took: 54.16 ms | cam=1.66 prep=1.18 pred=51.43
pred: 8.42 deg. took: 56.94 ms | cam=1.06 prep=1.32 pred=54.56
pred: 25.22 deg. took: 54.64 ms | cam=1.50 prep=1.14 pred=51.90
pred: [illegible] deg. took: [illegible] ms | cam=1.34 prep=1.44 pred=54.11
pred: [illegible] deg. took: [illegible] ms | cam=1.07 prep=1.12 pred=51.91
pred: 35.49 deg. took: 56.54 ms | cam=3.16 prep=1.22 pred=52.16
pred: 18.51 deg. took: 54.30 ms | cam=0.71 prep=1.33 pred=52.40
pred: 23.58 deg. took: 56.36 ms | cam=3.98 prep=1.51 pred=53.02
```

The code shown reads the frame from the ‘epoch-1.avi’ file and places the angle as text on the screen and the screen displays the frame and Angle as above. The q key acts as a breakaway as closing the window will not end the video and will reopen.

Conclusion:

For this entire semester, our end goal was to recreate a prototype of a self driving car given data and two boards. We started from simply making an LED blink on the board and made it all the way to this complex dual board system that recreates a miniature self driving vehicle. The project and subsequent labs have taught us a great deal about a multitude of things from sensor/actuator communication, Polling, UI, and board-to-board communication that allows us to do quite a bit of fun things. The knowledge that I have personally gained from this class will most definitely help me in my future career academically as well as my own personal projects with microcontrollers that I now feel much more confident that I can complete and make run effectively. Given where we started on this project at the beginning of the semester, I would say that we have all learned a great deal about embedded systems and how/when to use an embedded system and serial communication. In the future I plan to incorporate more embedded systems not only into my professional career but my personal life to control aspects of my own life.