

Lab 10

Luke Reicherter and Ethan Le

Task 1 - Objective

- Simulate auto-breaking detection using the LIDAR
- If distance is > 200 , display green LED
- If $200 \geq \text{distance} > 100$, display yellow LED
- If $100 \geq \text{distance} > 60$, display red LED
- If $60 \geq \text{distance}$, display blinking red LED

Task 1 - Our Implementation

- We first check if the “devid” is ready to receive data
- If so, we read the first two bytes to confirm if they are both ascii “Y”
- We then read the remaining bytes to obtain the distance
- Depending on the distance, varying LEDs will turn off/on

```
// Task-1:
// Your code here (Use Lab 02 - Lab 04 for reference)
// Use the directions given in the project document
if (ser_isready(devid)) {
    if ('Y' == ser_read(devid) && 'Y' == ser_read(devid)) {
        uint16_t dist = (ser_read(devid) | ser_read(devid) << 8);
        printf("%d\n", dist);

        if (dist > 200) {
            gpio_write(GREEN_LED, ON);
            gpio_write(RED_LED, OFF);
        }

        else if (dist > 100 && dist <= 200) {
            gpio_write(GREEN_LED, ON);
            gpio_write(RED_LED, ON);
        }

        else if (dist > 60 && dist <= 100) {
            gpio_write(GREEN_LED, OFF);
            gpio_write(RED_LED, ON);
        }

        else if (dist <= 60) {
            gpio_write(GREEN_LED, OFF);
            gpio_write(RED_LED, ON);
            delay(100);
            gpio_write(RED_LED, OFF);
        }
    }
}
```

Task 1 - Issues

- The LIDAR tends to read inaccurately
 - It is difficult to show each LED corresponding to distance
 - The range for these LEDs are extremely tight
- The position of the LIDAR can mess up the input

Task 2 - Objective

- Take angle measurements from the driving video
- Send the angle from the Raspberry Pi to the HiFive Board
- Receive the angle and return the degree back to main

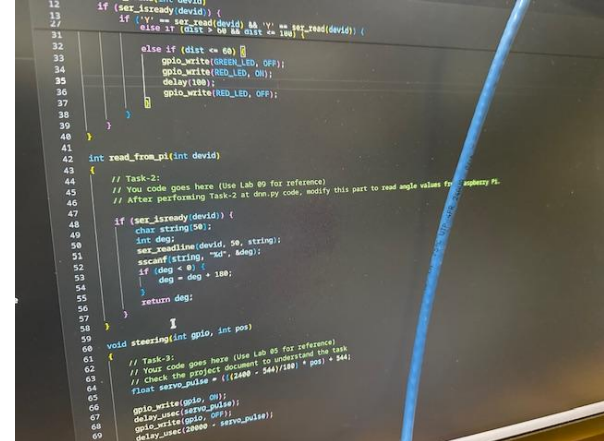
Task 2 - Our Implementation

- We first receive the angle from video and type cast it into an int, and then a string
- We then write the degree to the serial connection as a byte
- Using the c code on the HiFive board, we read the byte and store it in a string
- Using sscanf, we convert the string into an int
- We then return the int to main

```
pred_start = time.time()

#Feed the frame to the model and get the control output
rad = model.predict(img)[0][0]
deg = rad2deg(rad)

# Your code goes here in this if statement
# The if condition is used to send every 4th
# prediction from the model. This is so that
# the HiFive can run the other functions in between
if count%4 == 0:
    strDeg = str(int(deg)) + '\n'
    ser2.write(bytes(strDeg.encode()))
    |
```



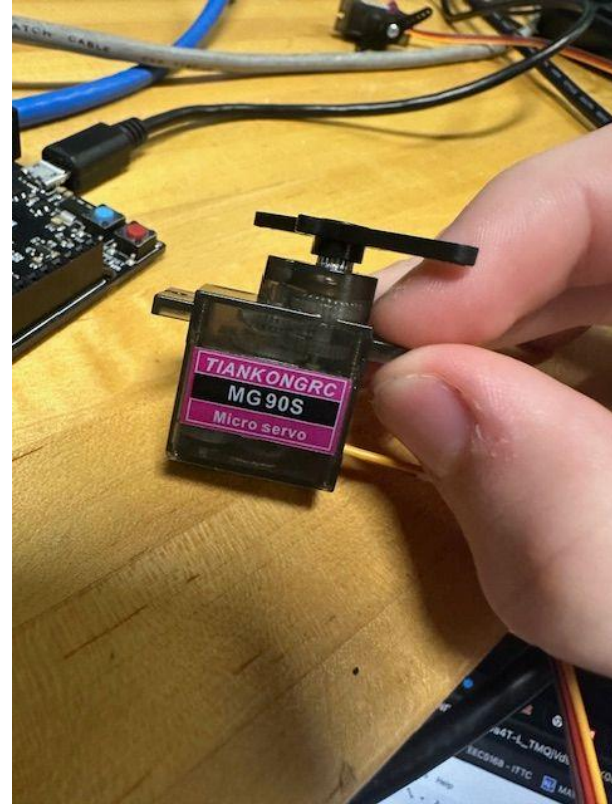
```
12 if (ser_isready(devicid)) {
13     if ('Y' == ser_read(devicid) && 'Y' == ser_read(devicid)) {
27         // You code goes here (Use Lab 09 for reference)
31         // After performing Task-2 at dnm.py code, modify this part to read angle values for raspberry pi.
32         char string[50];
33         ser_readline(devicid, 50, string);
34         sscanf(string, "%d", &deg);
35         if (deg < 0) {
36             deg = deg + 180;
37         }
38         return deg;
39     }
40 }
41
42 int read_from_pi(int devicid)
43 {
44     // Task-2:
45     // Your code goes here (Use Lab 09 for reference)
46     // After performing Task-2 at dnm.py code, modify this part to read angle values for raspberry pi.
47     if (ser_isready(devicid)) {
48         char string[50];
49         ser_readline(devicid, 50, string);
50         sscanf(string, "%d", &deg);
51         if (deg < 0) {
52             deg = deg + 180;
53         }
54         return deg;
55     }
56 }
57
58 void steering(int gpio, int pos)
59 {
60     // Task-3:
61     // Your code goes here (Use Lab 05 for reference)
62     // Check the project document to understand the task
63     //int servo_pulse = ((2000 - 544)/180) * pos + 544;
64     gpio_write(gpio, 0);
65     delay_us(servo_pulse);
66     gpio_write(gpio, 1);
67     delay_us(servo_pulse);
68     gpio_write(gpio, 0);
69     delay_us(servo_pulse);
70 }
```

Task 2 - Issues

- Establishing the connection between the Raspberry Pi and the HiFive Board
 - We did not receive any data for a good bit of time
 - This was fixed we encoded the degree properly
- Getting into the right directory
 - We didn't realise we were in the wrong directory for a good bit of time

Task 3 - Objective

- Use the received angle data from the previous task to turn the servo motor



Task 3 - Our Implementation

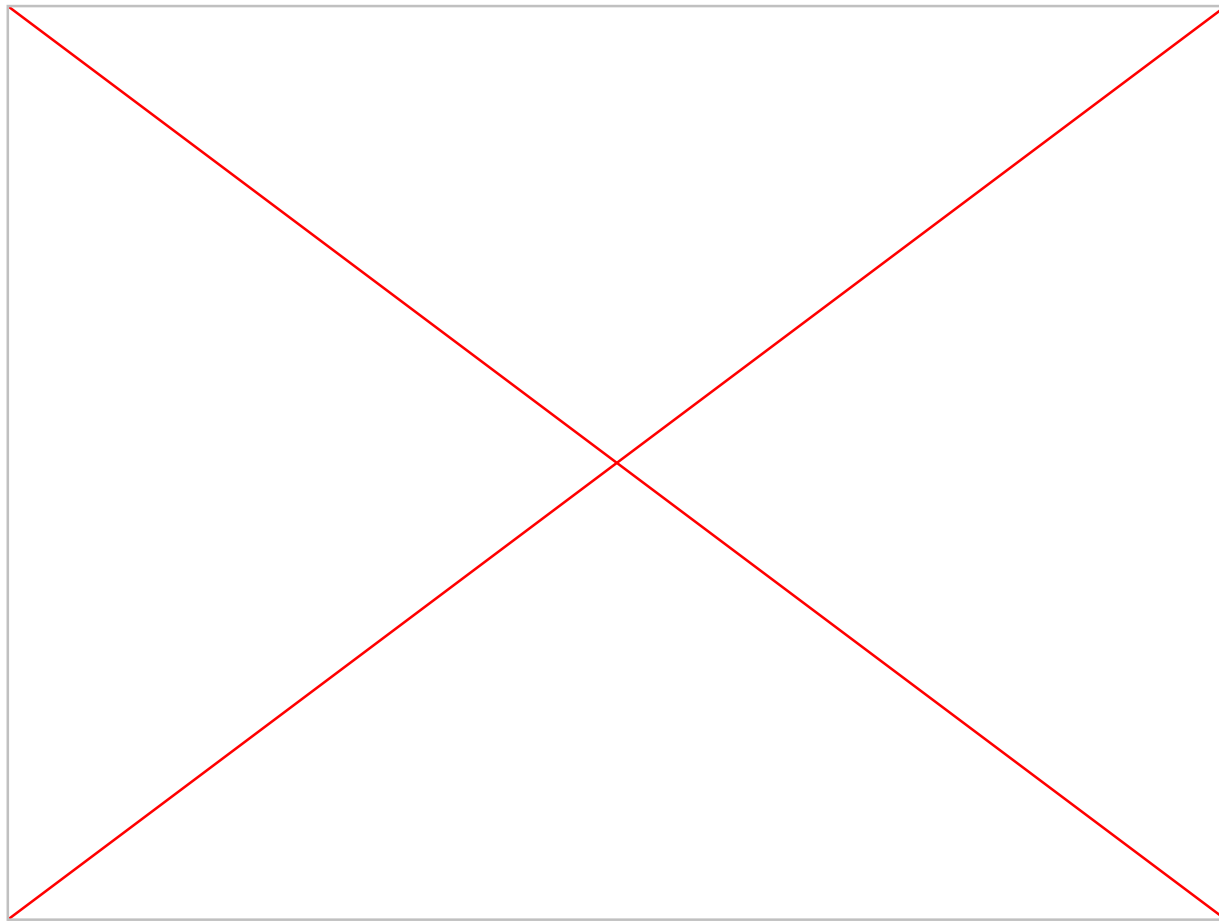
- We used a similar implementation as Lab 5
- We first calculated the servo pulse using:
 - $((\text{pulse_max} - \text{pulse_min}) / 180 * \text{angle}) + \text{pulse_min}$
- Turn on gpio connected to the servo motor
- Delay by the calculated pulse
- Turn off the gpio
- Delay by the remaining time in the period
- We do this to create the appropriate duty cycle for the input position

```
54     deg = deg + 180;
55     }
56     return deg;
57 }
58 }
59
60 void steering(int gpio, int pos)
61 {
62     // Task-3:
63     // Your code goes here (Use Lab 05 for reference)
64     // Check the project document to understand the task
65     float servo_pulse = (((2400 - 544)/180) * pos) + 544;
66
67     gpio_write(gpio, ON);
68     delay_usec(servo_pulse);
69     gpio_write(gpio, OFF);
70     delay_usec(20000 - servo_pulse);
71 }
72
73
74 int main()
75 {
76     // initialize UART channels
77     ser_setup(0); // uart0
78     ser_setup(1); // uart1
79     int pi_to_hifive = 1; //The connection with Pi uses uart 1
```

Task 3 - Issues

- Main Issue: Hardware
 - We ran the code multiple time only to receive no output from the servo motor
 - This was fixed by restarting the computer and changing out components

Demo Video



Future Improvements

- Take the time to write the entire code
 - This would help us gain a better understanding of the entire system
- Make sure to check for hardware related issues before rewriting code
 - This would have saved us a lot of time

Thank You For Listening