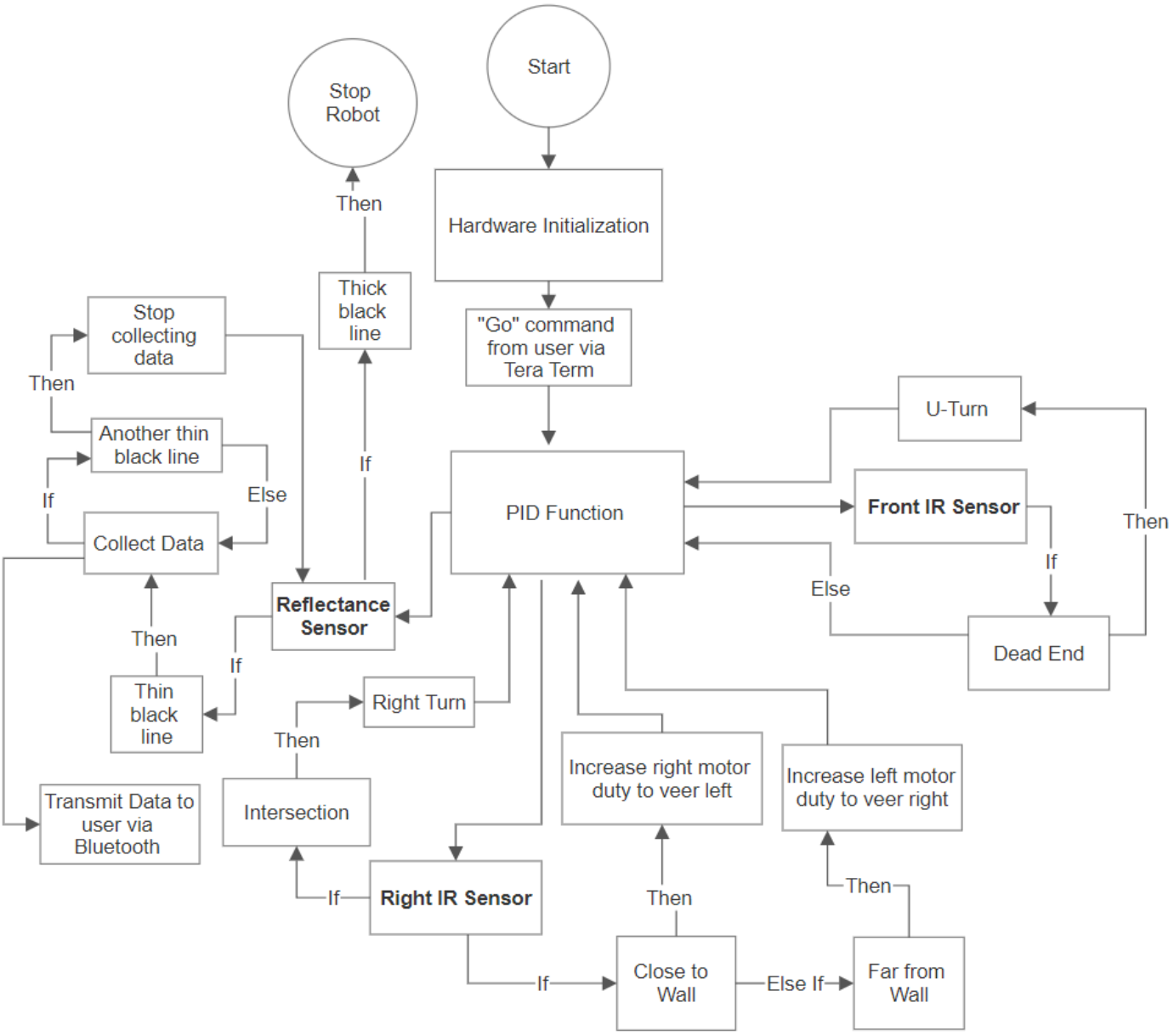
UNIVERSITY OF HOUSTON
CULLEN COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ECE 4437
Dr. Harry Le

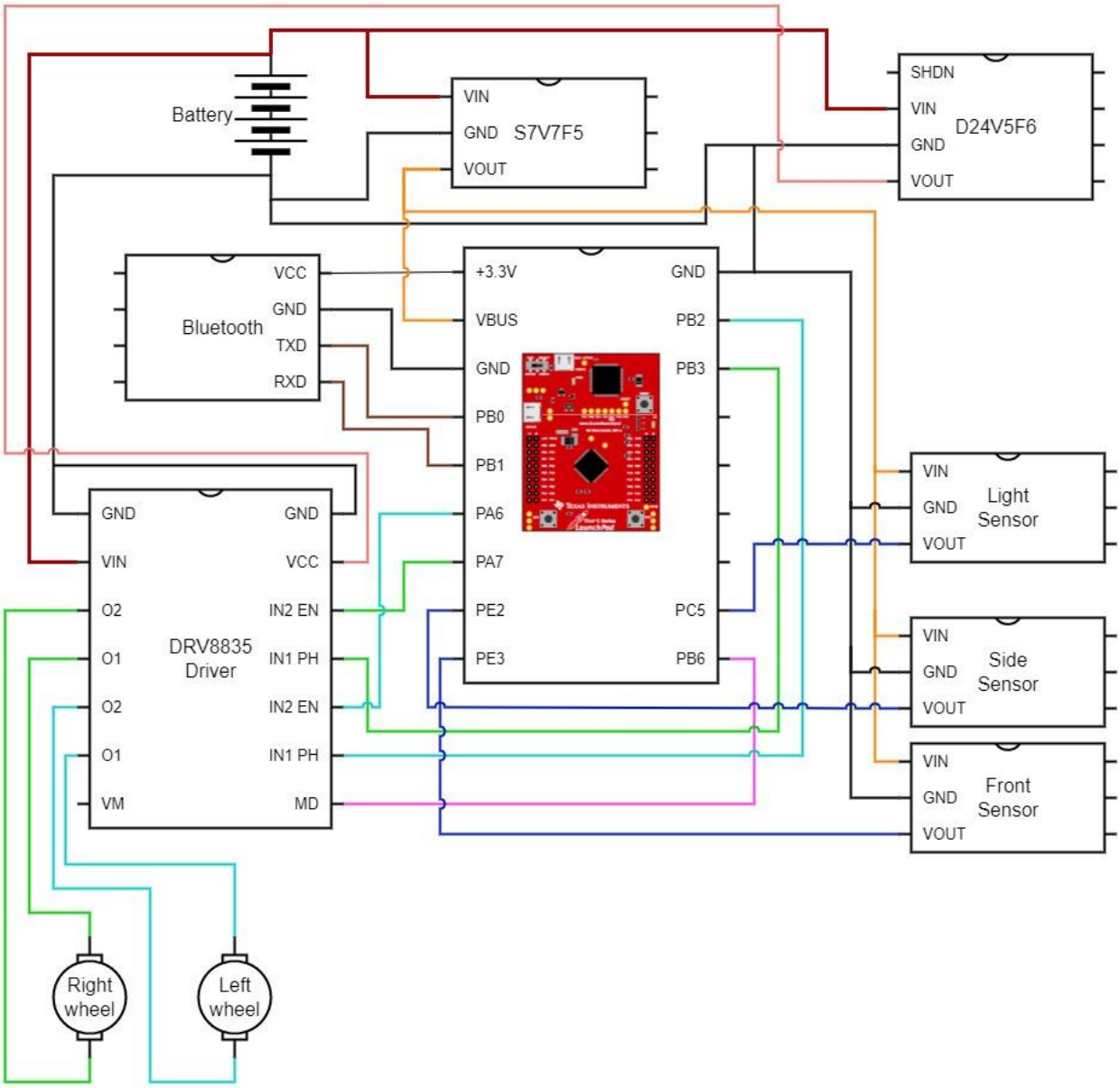
FINAL PROJECT REPORT

Team 17
Tuan Nguyen
Ryan Le
Kevin Tieu

I. Block Diagram



II. Wiring Diagram



III. Pseudocode and function description

1. Hardware initialization [Hardware_Init()]

Set the CPU clock to 40 MHz
Enable port F pin 1-3 as output pins

Enable UART1 and port B
Configure PB0 as RX, PB1 as TX
Set baud rate 9600, word length to 8 bits, no parity bits, and 1 stop bit
Enable UART1 interrupt for TX and RX

Enable port E for side (PE2) and front distance sensor (PE3)
Configure ADC0 for side distance sensor using sample sequence number 3, channel 0.
Configure ADC1 for front distance sensor using sample sequence number 3, channel 1.

Enable port B pin 2, 3, 6 as outputs for motor driver
Set PB6 to choose Drive/Brake mode for the motor

Enable PWM Module 1 using PA6 and PA7
Configure PA6 to use Module 1 PWM2 in counting down mode
Configure PA7 to use Module 1 PWM3 in counting down mode
Set the maximum PWM pulse width to 6250 for both PWM2 and PWM3

Enable timer0, timer1, and timer2
Set timer2 to periodic, period of 50 ms
Enable interrupt on timer2 when time out
Configure timer0 and timer1 to periodic and count up

2. LED control functions [offLED(), redLED(), blueLED(), greenLED(), whiteLED()]

a. offLED()

Clear PF1-PF3
Print→ “Tiva LED = off”

b. redLED(), blueLED(), whiteLED(), greenLED()

Turn off and on the LED of the respective color (red, blue, green and white)
Print the message to the Tera Term.

3. UART message [getMsg()]

If no character in buffer:
    Return
Get 2 characters from buffer

Search lookup table for the function corresponds to the command
If found:
    Execute the function

4. **Distance sensor functions [adc_read_side, adc_read_front, current_distance]**
    a. **adc_read_side() [Read ADC reading from side sensor]**

Clear ADC1 interrupt
Trigger ADC sampling
Get ADC reading and store in adc_side

    b. **adc_read_front() [Read ADC reading from front sensor]**

Clear ADC0 interrupt
Trigger ADC sampling
Get ADC reading and store in adc_front

    c. **current_distance(char c) [Convert ADC reading to distance in cm]**

If c = ‘f’ :
    Get ADC value of front distance sensor
If c = ‘s’ :
    Get ADC value of side distance sensor
If ADC > 100:
    Voltage = 3.3 * ADC / 4095
Else
    Voltage = 3.3
Distance = 13 * (Voltage)^(-1.1)
Return Distance

    d. **Rgb_switch(Distance) [Change LED based on distance]**

If Distance <= 4 cm
    red LED
Else if 4 cm < Distance <= 7 cm
    yellow LED
Else
    green LED

5. **Motor functions [goForward(), goReverse(), stop(), Turn()]**
    a. **goForward()**

Enable PWM2, PWM3
Write 0 to PB2, PB3
Print → “Full Speed Forward”

b. goReverse()

Enable PWM2, PWM3
Write 1 to PB2, PB3
Print → “Full Speed Reverse”

c. stop()

Disable PWM2, PWM3
Print → “Motor Off”

d. Turn()

Argument for function is ‘l’ for left and ‘r’ for right
If ‘l’- Write 1 to left wheel to reverse and write 0 to right wheel to forward reverse left wheel
If ‘r’- Write 1 to right wheel to reverse and write 0 to left wheel to forward reverse left wheel

e. swi_Turning [Swi_Turn] (software interrupt)

Use to turn left or u-turn when an obstacle is in front of the robot

While (adc_front > 750):
    Turn Left
    Read_adc_side
    Read_adc_front

6. PID() [Follow the wall using PID]

Uses PID calculation based off of ADC values to obtain error from setpoint of side sensor updated continuously. This algorithm will be used to adjust the motors to ensure the robot stays on a straight path.

LightSensor()
PreviousError = 0
Sum = 0
Turn off LED
adc_read_side() → adc_side
adc_read_front() → adc_front
ErrorSide = 1800 - adc_side
Current_side = current_distance(‘s’)
Setpoint = 13*(3.3*1800/4095)^(-1.1)
Error = Setpoint - Current_side
P = 0.06 * ErrorSide
Sum = Sum + ErrorSide

```
I = 0.001 * Sum
D = 0.0001 * (ErrorSide - PreviousError)
PWM = | P + I + D |
duty1 = duty - PWM
If duty1 < 10:
        duty1 = 50
If ErrorSide > 0:
        Right wheel ← Speed {duty1}%
        Left wheel ← Speed 80%
Else if ErrorSide < 0:
        Right wheel ← Speed 80%
        Left wheel ← Speed {duty1}%
If adc_front > 2250:
        Turn()
If ( i % 2 = 0 & Data_collect):
        blueLED()
        If (countPing < 20 & ping_sent = false):
                Ping→Append(Error)
                countPing++
        Else if (countPong < 20 & pong_sent = false):
                Pong→Append(Error)
                countPong++
If (countPing = 20 || countPong = 20):
        send_buffer()
Turn off green LED
Increment i
```

7. **LightSensor() [Read light sensor]**

Used to indicate first, second, and thick black line. If the first thin line is detected, data collect will be true and start the PID algorithm. Data collection and printing to terminal will continue until the second black line. This will stop data collection and send the remaining data to be output to the terminal. Once the robot reaches a thick black line, it will stop completely.

```
Read and write into a GPIO pin
Establish start timer variable
While pin and GPIO port is active
        Read GPIO pin

Establish end timer variable
```

If end time - start time > 20000
    Print→"Sensor 2 is reading BLACK, ticks elapsed = (end time - start time)
    Set data collection to true
    Add one to counter
Else
    If counter >0 && counter < 4
        Set data collection to true
        Add one to thin black line counter
if counter >= 4
    Finish line is true
    Set data collection to false
    Set overwrite buffer to true

Reset counter to 0

If thin black line counter == 2 and Data_collect
    Print → “Second Line”
    Send data and print to terminal
Else if finish line is detected:
    Print→"Finish Line Reached, Time Elapsed: (PID counter) seconds”
    Stop motor movement

8. **Send_buffer [Send errors to UART]**

Turn on green led to signify that data is being transmitted and print out error values from ping and pong arrays into bluetooth terminal in the form of ASCII hex

If ping array==20 and ping not sent
    Send ping modbus
    Reset ping array

Else if pong array==20 and pong not sent
    Send pong modbus
    Reset pong array

9. **Hardware interrupts**
    a. **Bluetooth_UART**

Get UART interrupt status
Clear UART interrupt
getMsg()

    b. **Timer2_50ms**

Clear timer2 interrupt
If the motor is on || “go” command:
    PID()