

Project 2

CSCE 336 Embedded Systems, Spring 2024

Project 2

Checkpoint 1 – “IR Decoding” Thursday Apr 18th (in Lab)

Checkpoint 2 – “Robot Motion with IR Remote”, Tuesday, Apr 30th (in Lab)

A-Functionality – “Wall Following & Obstacle Avoidance”, Thursday, May 9th (in Class and Lab)

Competition, Thursday, May 9th (in Class and Lab)

Code and Written Report: Friday, May 10th (Close of Business)

Name: _____

1. Instructions

This Project is an individual assignment, collaboration is not allowed. Show your work and describe your reasoning to get partial credit if your solution is incorrect. Complete all of the sections below and make sure to get the instructor or TA to sign off on the cutsheet where required. You should keep notes on what you complete since it will be included in the project report.

All code will be uploaded in your code project folder on Bitbucket! Failing to electronically turn in your code will result in up to at least a 20 point penalty on this assignment. Points may also be deducted for coding errors, poor style, or poor commenting.

2. Late Work

If problems arise with graded assignments, see your instructor in advance. Assignments portions turned in or checked off later than the due date without prior permission from the instructor will be penalized 10% per day on that portion. Plan ahead to get things completed on time.

3. Robot Assembly with IR Receiver

Connect your IR receiver to your assembled robot. Insert the IR receiver module into the protoboard. Use M/F wires to connect your protoboard to your Arduino Uno.

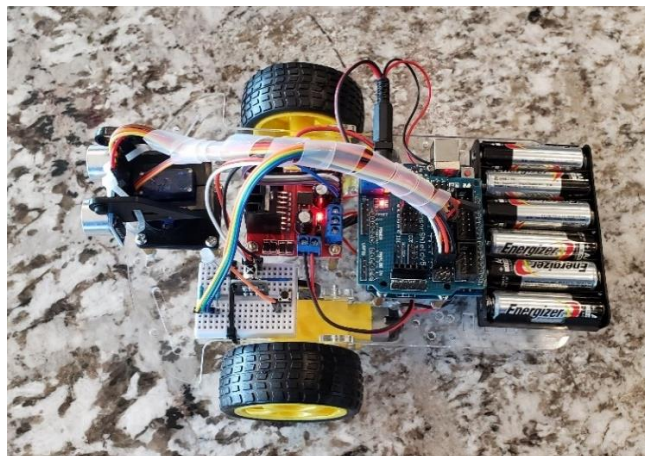


Figure 1: Assembled Robot with IR Receiver

Project 2

You will use a regular wire to connect your signal and ground to the logic analyzer or oscilloscope to test the IR signal.

When you are looking at the sensor ball on your IR receiver module, the pin on the left is your signal pin; the pin in the middle is your ground pin; and the pin on the right is your Vcc. Note: On the IR Receiver Module the Vcc and ground are flipped. See figure 1 and 2 below.

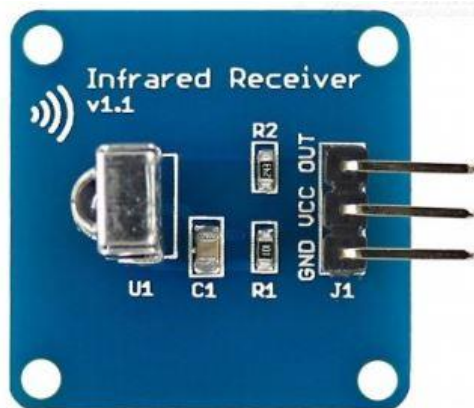
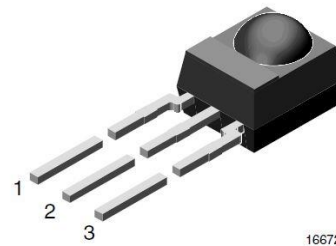


Figure 2: IR Receiver Module



MECHANICAL DATA

Pinning:

1 = OUT, 2 = GND, 3 = V_S

Figure 3: IR Receiver Sensor

On your Arduino Uno, connect the signal pin to the External Interrupt 0 (INT0) pin on Port D, pin 2 and then connect the ground and Vcc to the respective pins on the Arduino.

4. IR Decoding and Timer Counts

Download the test.ino file on the website. Build a project around your modified test.ino and then add the required lines of code to enable the INT0 pin and Timer 1 so that you can count the pulse widths on the IR Sensor. An input will look something like the waveform shown in Figure 4 below.

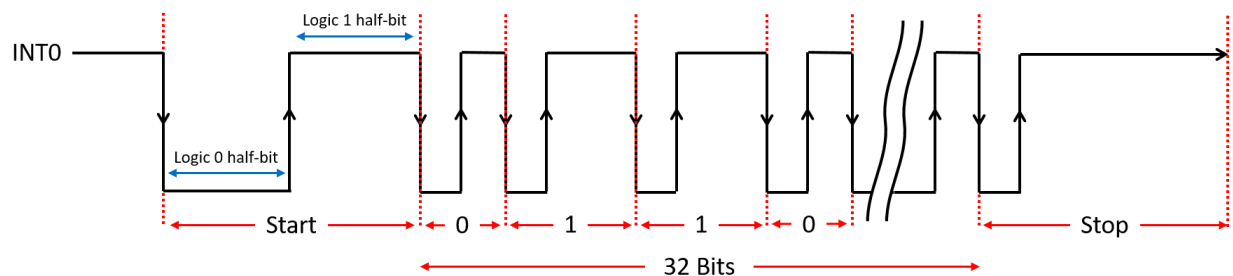


Figure 4: IR receiver signal showing all bits

Once you enter the required lines of code you should get an output that looks something like the following in Figure 5 below.

Project 2

```
test | Arduino 1.8.8
File Edit Sketch Tools Help
test $
38 Serial.println("Waiting for IR input");
39
40 while(IR_DECODER_PIN != 0); // IR input is nominally logic 1
41 TCNT1 = 0; // reset timer
42 newIrPacket = FALSE;
43 // Clear Timer/Counter1, Output Compare A Match Flag by writing 1
44 // Enable interrupt on match with OCR1A
45
46 for(i=0; i<SAMPLE_SIZE; i++) {
47
48   TCNT1 = 0; // reset timer and
49   while(IR_DECODER_PIN==0 && newIrPacket == FALSE); // wait while IR is logic 0
50   time0[i] = TCNT1; // and store timer 1 count
51
52   TCNT1 = 0; // reset timer and
53   while(IR_DECODER_PIN != 0 && newIrPacket == FALSE); // wait while IR is logic 1
54   time1[i] = TCNT1; // and store timer 1 count
55
56 } // end for
```

```
Starting up.
Waiting for IR input
Time Logic 1
time1[0] = 1133
time1[1] = 432
time1[2] = 433
time1[3] = 432
time1[4] = 156
time1[5] = 155
time1[6] = 155
time1[7] = 155
time1[8] = 156
time1[9] = 432
time1[10] = 432
time1[11] = 432
time1[12] = 155
time1[13] = 155
time1[14] = 149
time1[15] = 155
time1[16] = 155
time1[17] = 155
time1[18] = 155
time1[19] = 156
time1[20] = 155
time1[21] = 155
time1[22] = 432
time1[23] = 427
time1[24] = 155
time1[25] = 433
time1[26] = 433
time1[27] = 432
time1[28] = 427
time1[29] = 426
time1[30] = 155
time1[31] = 149
time1[32] = 426
time1[33] = 0
Time Logic 0
```

Figure 5: Test.ino code and sample output on serial port

A few questions about what goes on in this program are in order before we proceed.

- 1) How long will it take the timer to roll over?
- 2) How long does each timer count last?

The loop in the test sketch file reads in the IR pulse in the for loop.

Annotate the picture below in Figure 6 to indicate which line of the for loop in the program is executed at which part of the pulse. You should show a total of 6 lines of code (lines 48-51 and lines 52-54).

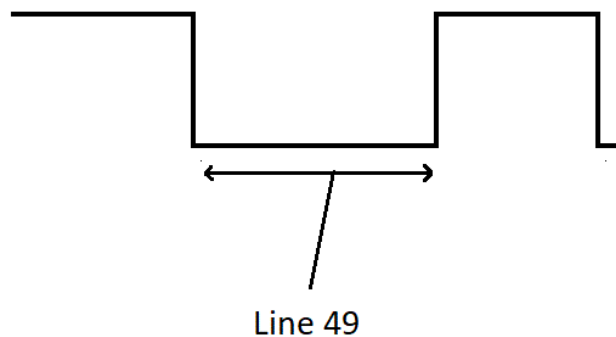


Figure 6: IR receiver signal relating to code

Project 2

IR Data Packets

Before you start on this portion of the assignment, watch [Dave Jones' Trigger Hold-off Tutorial](#). You are going to need to use the logic analyzer or an oscilloscope to examine the IR waveforms generated by a remote control of your choice.

Set up your robot like the picture in below. Make sure to connect the power and ground in the correct order! Connect the logic analyzer on the Vout pin of the IR receiver.

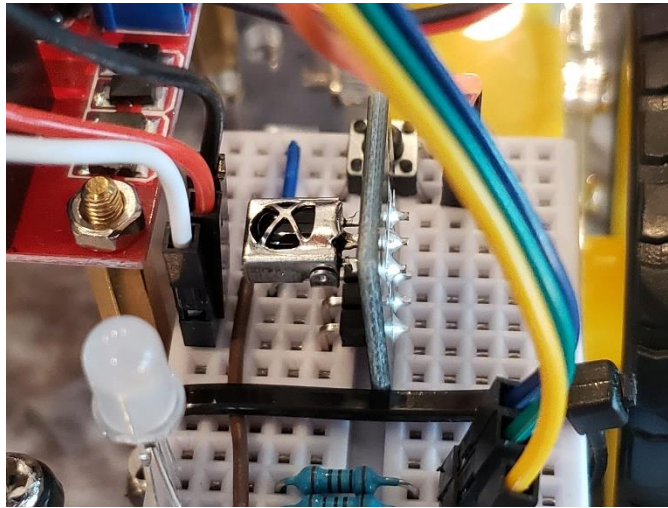


Figure 7: IR receiver signal showing

Configure the logic analyzer to collect data on an edge change, with at least 90% of the data stored post-trigger. On my remote control, the full remote control signal was about 60ms. Please note that remote control data packets are not standardized by any means, so the remote that you use to perform these experiments will almost certainly generate results different from those that your neighbor's remote will generate. Additionally you will need to

List the lengths of the pulses generated by the remote control in absolute time using the logic analyzer (3 significant figures) and in timer A counts. Note: "start -- logic 0 half pulse" refers to the logic LOW portion of the start pulse, and "data 0 -- logic 1 half pulse" refers to the second half (which is a logic HIGH) of the pulse representing a zero bit.

Pulse	Duration (ms)	Timer 1 counts(dec)	Timer 1 counts(hex)
Start -- logic 0 half pulse			
Start -- logic 1 half pulse			
Data 1 -- logic 0 half pulse			
Data 1 -- logic 1 half pulse			
Data 0 -- logic 0 half pulse			

Project 2

Data 0 -- logic 1 half pulse			
Stop -- logic 0 half pulse			
Stop -- logic 1 half pulse			

Collect and tabulate in Excel eight samples of timer 1 counts for each of the following pulse types (in decimal). Compute the average and standard deviation of each pulse type. I would suggest just grabbing it from the serial output.

Pulse	Average	Std Dev
Data 1 -- logic 1 half pulse		
Data 0/1 -- logic 0 half pulse		
Data 0 -- logic 1 half pulse		

Ensure you label the rows and columns of your table so that I will know what the information in each cell means.

For each pulse type list the range of timer 1 counts that would correctly classify 99.9999426697% of the pulses. This number has something to do with the [standard deviation](#) (hint: look at the table in this section).

Pulse	Lower Range	Upper Range
Data 1 -- logic 1 half pulse		
Data 0/1 -- logic 0 half pulse		
Data 0 -- logic 1 half pulse		

Write the codes (in hex) for at least 10 remote control buttons. ("Button name" refers to CH+, 6, VOL-, Power, etc.)

Number	Remote Button	Hex Code
1		
2		
3		
4		
5		
6		

Project 2

7		
8		
9		
10		

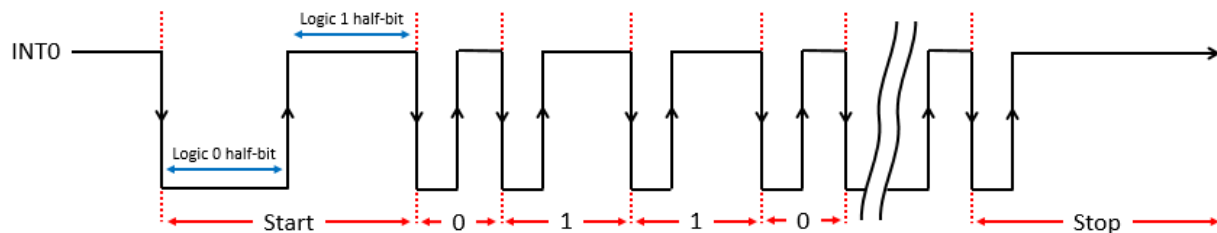
Checkoff: Show the instructor your picture of from the logic analyzer (or oscilloscope), tables of logic half pulse counts, and hex codes for your IR remote. This will all be in your report as well.

Instructor will sign off using Canvas Quiz submission.

5. “Robot Motion with IR Remote”

Overview:

Using the project2.ino, project2.cpp, and project2.h files you will program your robot to read the IR pulses from an IR remote and decode them for use with your robot. In addition to the starter sample code you will use the pseudocode given in the following image in figure 8 to finish the code to decode the button presses. Once you have the IR remote button presses decoded you can use them to control your robot. You will use the buttons on a remote control for Robot Motion (up, down, left, right, stop) on you robot. Make the robot move forward, backward, left turn, right turn, change speed.



<p>Main Code:</p> <ul style="list-style-type: none"> IR_Decoder_Setup Enable Global Interrupts Poll newIRPacket 	<p>INT0_Vect (↓ Edge):</p> <ul style="list-style-type: none"> Read Timer 1 Register count Classify Logic 1 half-pulse & shift into irPacket Store count in packetData[] Disable T1 interrupt on match Enable ↑ edge interrupt 	<p>INT0_Vector (↑ Edge):</p> <ul style="list-style-type: none"> Clear Timer 1 count Clear T1 output comp match flag Enable T1 interrupt on match Enable ↓ edge interrupt
<p>IR Decoder Setup:</p> <ul style="list-style-type: none"> Set Timer 1 in CTC mode with OCR1A as top Set prescaler to clk/64 Clear Timer 1 count Set INT0 pin as input Enable INT0 pin pullup resistor Enable ↓ edge INT0 Interrupt 	<p>Constants:</p> <pre>#define IR_DECODER_PIN (PIND & (1 << PIND2)) #define IR_PIN (PIND2) #define HIGH_2_LOW (EICRA = (EICRA & ~(1 << ISC00)) (1 << ISC01)) #define LOW_2_HIGH (EICRA = (1 << ISC00) (1 << ISC01)) #define Up_Button 0xE0E006F9 #define Down_Button 0xE0E08679 #define Left_Button 0xE0E0A659 #define Right_Button 0xE0E046B9</pre>	
<p>Global Variables:</p> <ul style="list-style-type: none"> uint32_t irPacket; uint8_t newIrPacket = FALSE; uint16_t packetData[34]; uint8_t packetIndex = 0; 	<p>Timer1_CompA_Vect</p> <ul style="list-style-type: none"> Disable T1 interrupt on match Clear packetIndex Set newIRPacket flag 	

Figure 8: IR receiver pseudocode and waveform

Required Functionality “Robot Motion” - Checkpoint 2

Demonstrate that you can use your IR Remote to make your robot move forward, backward, stop, turn left, turn right, and be able to vary the speeds. The robot should perform these movements while completely disconnected from a computer (i.e. no USB cord).

Checkoff: *Demonstrate movement forward, backward, left turn, right turn, and vary the speed of the motors. The robot should perform these movements while completely disconnected from a computer (i.e. no USB cord)?*

Post a link to a video using the Canvas Quiz for checkoff.

A Functionality “Wall Following and Obstacle Avoidance”

Using your ultrasonic sensor and servo demonstrate that your robot can follow a wall (i.e. staying within a foot of the wall without touching it) and navigate around two obstacles based on the obstacle course shown in Figure 9.

Checkoff: *Demonstrate that your robot navigate around the two obstacles and along the wall for 20 feet (i.e. staying within a foot of the wall without touching the wall or the obstacles).*

Post a link to a video using the Canvas Quiz for checkoff.

Competition Bonus Scoring

For the Competition Bonus the following scoring rules will apply:

- Must compete in competition on the day of the event.
- You get three attempts to compete in the event.
- You must use the ultrasonic sensor and servo to navigate around the obstacles.
- There are 4 marked points on the course. Each point reached will result in 1 point and finishing will result in 4 points. This means you get 4 points for finishing the course.
 - You will be docked points for contact with the wall and the obstacle.
- The fastest time will get 6 additional points, second 5 points, and third 4 points. Those finishing within the top third of the class (excluding top three) will get 2 points, second third will get 1 additional point.

Project 2

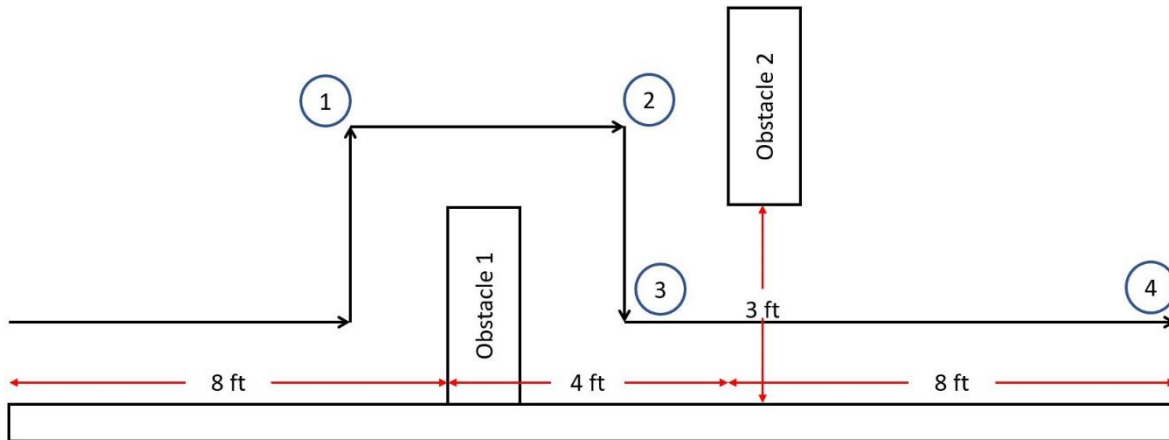


Figure 9: Project 2 Competition Course

Report

Don't forget to include your hardware schematic! Software flowcharts and debugging information are also very useful. Finally, you'll also want to document which remote control you used.

Be sure to include appropriate (not blurry or in need of rotation) screenshots, pictures, and tables. Do not include anything handwritten without prior approval.

Finally, ensure you are committing regularly.