

Remote Motor Control with Feedback

1 Design Objectives

- Receive and interpret wireless data packets.
- Learn about quadrature rotary encoders.
- Configure GPIO pin(s) as interrupts.
- Derive a relation between the encoder outputs and distance traveled.

2 Wireless Data Transmission

The first part of this lab will have you load a pre-compiled binary image on the ECE353 controller board that will send data to the ECE315 robot. You will use a provided library to read the data packets sent by the ECE353 controller board. The information in the data packets will then be used to control the direction and speed of the robot by calling your PWM functions developed in Lab 2.

The compiled binary image for the ECE353 carrier will allow you to control your robot using the PS2 joystick. The first time that you load the image onto the ECE353 controller card, you will need to supply the controller board with two unique IDs that will allow the controller board to communicate only with your robot. **The IDs can be set using the serial debug interface on the ECE353 board and will be written to the on-board EEPROM.**

2.1 Programming the ECE353 Board to be the Robot Controller Board.

1. Connect the USB cable for the ECE353 controller board and turn the board on.
2. Download Lab3-Files.zip from the course website. Extract the files to your PC
3. Open the project file in ECE353-Robot-Controller directory.
4. Download the executable image to the board using the Load button. **DO NOT** recompile the project or the executable will be lost.



5. Open a Putty terminal at a connection speed of 115200
6. Hit the Reset Button

E C E 315: Introductory Microprocessor Laboratory

7. If the controller boards IDs are not set, you will be prompted to enter the last two digits of the ECE353 controller boards ID and the ECE315 robot's ID.

The format of the IDs must be as follows

ECE353 Controller: 0x353XX

ECE315 Robot: 0x315YY

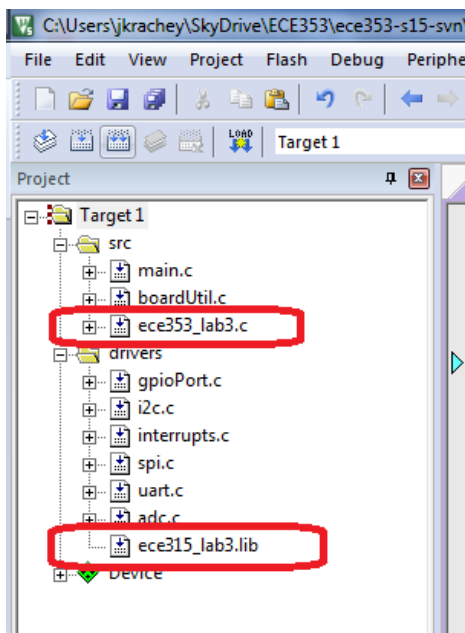
If the controller board's IDs are already set or if you need to set the controller board's IDs to a different values, press the UP directional button. You will be prompted for new IDs. Once the IDs are set, press the reset button.

The TA will assign you the values to use for XX and YY. Once the IDs have been set, you will need to press the reset button on the controller board.

Make note of the two IDs that you have set. You will need these values when writing the code for your ECE315 robot.

2.2 Installing the Lab 3 Libraries In the ECE315 Base Project

8. Copy ece315_lab3.c and ece315_lab3.h into the ECE315 Directory
9. Double Click on the src folder in the Project pane and add ece315_lab3.c to the project.
10. Copy ece315_lab3.lib into the ECE315 directory
11. Double Click on the drivers folder and add the ece315_lab3.lib to your project. Make sure to change the 'Files of Type' to be Library File
12. Copy wireless.h into the include directory



13. In main.c, include ece353_lab3.h in man.c

```
30
31
32 #include "TM4C123.h"
33 #include "boardUtil.h"
34 #include "ece353_lab3.h"
35
```

E C E 315: Introductory Microprocessor Laboratory

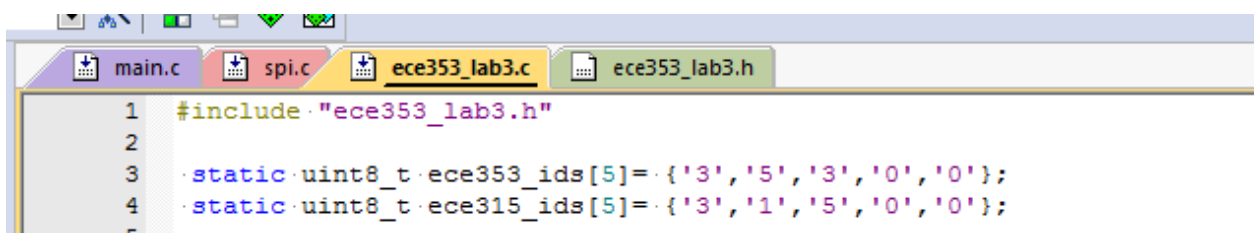
14. In `main.c`, call `rfInit()` from the `initializeBoard` function

```
43 //*****
44 //*****
45 void initializeBoard(void)
46 {
47     DisableInterrupts();
48     serialDebugInit();
49     rfInit();
50     EnableInterrupts();
51 }
52
```

15. In `main.c`, add the following lines of code to the `main` function. `wireless_get_32()` will be used to receive data from the ECE353 controller board. The first parameter is a Boolean variable that indicates if the function should **busy wait** until data has arrived. Pass **true** to busy wait, pass **false** to return immediately. When data has been received, a return code `NRF24L01_RX_SUCCESS` will be returned.

```
53
54 //*****
55 //*****
56 int
57 main(void)
58 {
59     char msg[80];
60     wireless_com_status_t status;
61     uint32_t data;
62
63     initializeBoard();
64
65     uartTxPoll(UART0_BASE, "\n\r");
66     uartTxPoll(UART0_BASE, "*****\n\r");
67     uartTxPoll(UART0_BASE, "ECE315 Default Project\n\r");
68     uartTxPoll(UART0_BASE, "*****\n\r");
69
70     // Infinite Loop
71     while(1)
72     {
73         // Check to see when wireless data arrives
74
75         status = wireless_get_32(false, &data);
76         if(status == NRF24L01_RX_SUCCESS)
77         {
78             memset(msg, 0, 80);
79             sprintf(msg, "Data RXed: %c%c%d\n\r", data >> 24, data >> 16, data & 0xFFFF);
80             uartTxPoll(UART0_BASE, msg);
81         }
82     }
83 }
84
85
```

16. Modify `ece315_lab3.c` so that `ece353_ids` and `ece315_ids` match the values you entered on the ECE353 controller board's serial terminal.



```
1 #include "ece353_lab3.h"
2
3 static uint8_t ece353_ids[5] = {'3', '5', '3', '0', '0'};
4 static uint8_t ece315_ids[5] = {'3', '1', '5', '0', '0'};
5
```

E C E 315: Introductory Microprocessor Laboratory

17. Compile and load the resulting code onto your ECE315 robot.
18. Open a second serial terminal that is connected to the ECE315 robot @ 115200 baud.
19. Press the reset button on both boards
20. You should be able to observe that the ECE315 robot is receiving data from the ECE353 controller board by moving the PS2 Joystick.

[illegible]

E C E 315: Introductory Microprocessor Laboratory

2.3 Wireless Packet Format

Data packets will be transmitted as 4 bytes of data. The upper 16-bits are used to indicate direction. The lower 16 bits are used to indicate the duty cycle of the PWM peripherals used to control the motors. The characters for the upper 16 bits are ASCII characters.

Command	Bits 31-24	Bits 23-16	Bits 15-0
Forward	'F'	'W'	Duty Cycle
Reverse	'R'	'V'	Duty Cycle
Right	'R'	'T'	Duty Cycle
Left	'L'	'F'	Duty Cycle
Stop	'S'	'T'	Don't Care

2.4 Integration with PWM peripheral

Using the data packets that you receive from the ECE353 controller, use your PWM peripheral drivers from lab 2 to control the direction and speed of the robot.

3 Encoder Background

[Quadrature encoders](#) are widely used in the industry to estimate position, velocity and direction of rotation of a motor. The estimated parameters are then typically used in feedback control loops for controlling a process. In this application, the information from encoders will be used to accurately move the robot to specified distances. The ECE315 robot has two brushed DC motors each equipped with a quadrature encoder.

Each quadrature encoder has two output channels, namely `ENCODER_xx_A` & `ENCODER_xx_B` (where 'xx' denotes Left / Right). These channels are connected to GPIOs. Refer the robot's schematics to determine the correct port pins.

The number of pulses and phase relationship between the two channels can help us estimate the motor shaft position and rotation direction respectively. The animation in [this link](#) should give a better idea about the channel waveforms and its relationship with each other.

Estimating position: By considering the number of pulses for 1 complete revolution and the number of revolutions present in a known distance, the following formula can be derived:

$$\frac{\text{Number of pulses}}{\text{inch}} = \left(\frac{\text{Number of pulses}}{\text{known distance}} \right) * \left(\frac{\text{known distance}}{\text{inch}} \right)$$

Once the "number of pulses / inch" is determined, the software can be programmed to make the robot move to any particular distance by counting the required number of pulses.

E C E 315: Introductory Microprocessor Laboratory

3.1 GPIO Configuration

Configure external interrupts on the GPIO pins connected to the encoder channels. Follow the steps below to determine the number of pulses / inch.

- 1) Declare a function in `boardUtil.h` called `void encodersInit(void);`
- 2) Complete the code for `encodersInit` in `boardUtil.c` so that all 4 of the GPIO pins connected to the rotary encoders generate interrupts on rising and falling edges.
- 3) Write a GPIO ISR(s) in `interrupts.c` so that it counts the number of pulses on both the encoder channels separately.
- 4) NOTE: The encoder counts for all four GPIO pins should match if the encoders were reliable. On our robot, the encoders have shown to be unreliable. As a result, the two GPIO pins that make up an encoder pair will interrupt at different rates. This means that we will not be able to determine the direction the motors are moving.

We can however count the number of pulses on a single encoder GPIO pin to estimate distance. Drive the robot manually using the ECE353 controller board for 10 feet and output the number of pulses recorded in for each encoder GPIO pin.

(Note: It is crucial that this step is performed as accurately as possible so as to obtain the correct number of pulses. Repeat several times and obtain an average count).

- 5) Using the encoder that provides the most consistent results, calculate the number of pulses / inch using the formula provided above.

3.2 Feedback Loop Implementation

Write a subroutine in a C file you create called `encoders.c` that computes the number of pulses required to accumulate for a given distance. This value can then be decremented in the ISR and polled in the main routine until it reaches 0.

3.3 What to Demo and Deliver

- 1) Drive the ECE315 robot using the ECE353 controller board's PS2 Joystick.
- 2) Display the output from an encoder pair on the oscilloscope. Include a screen-capture in your deliverable.
- 3) Provide the IRQ counts for each of the following distances: 10 inches, 30 inches, and 100 inches.