UNIVERSITY OF COLORADO BOULDER

ECEN 2270: ELECTRONICS DESIGN LAB FINAL REPORT

# Movement Controlled Robot

Joseph Blake
Matthew Haney

Rachel Lawson
Milica Notaros

May 3, 2015

# Contents

**1 Introduction and Objectives** **2**

**2 Equipment** **2**

**3 Methods and Results** **3**

3.1 Control Input Design and Testing . . . . . . . . . . . . . . . . . . . . . . . . 3

3.1.1 Glove Controller Design . . . . . . . . . . . . . . . . . . . . . . . . . . 3

3.1.2 Muscle Sensor Testing . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

3.1.3 Flex Sensor and Accelerometer Testing . . . . . . . . . . . . . . . . . 7

3.1.4 Push Button and Potentiometer Implementation . . . . . . . . . . 8

3.2 Wireless Connection and Final Implementation . . . . . . . . . . . . . . . . 8

3.2.1 Wireless Setup and Code . . . . . . . . . . . . . . . . . . . . . . . . . 8

3.2.2 Packaging . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11

**4 Further Explorations** **11**

**5 Conclusion** **12**

**6 References** **12**

## 1 Introduction and Objectives

For our group's final project we added an additional custom component to our robot that incorporated both hardware and software. Our project consisted of a glove that controls the direction the robot moves, and a muscle sensor that controls the speed of the robot. The glove has two flex sensors and an accelerometer which combined will drive the robot forward, backwards, left, and right. We also have a muscle sensor that controls the speed by sensing the magnitude of the electrical activity of the muscle. There are three ranges of speeds: fast, medium, and slow. These components communicate wirelessly over WiFi. Overall, This project can be loosely divided into three parts: an array of inputs, a wireless transmitter that encodes the input data into a command, and a wireless receiver that translates the command into an action taken by the robot.

## 2 Equipment

Hardware:

- EMG Sensor (Sparkfun Muscle Sensor v3)
- Arduino UNO R3
- 2x Flex Sensor (FS7548)
- Accelerometer (ADXL337)
- Teensy 3.1 (PJRC)
- TeensyLC (PJRC)
- 2x Wireless Transceiver (ESP8266)
- Gardening Glove, Duct Tape
- Slide Potentiometer
- Push Button
- AA Batteries
- Robot with lab 1-4 components
- Agilent DSO1024A Oscilloscope

- Agilent E3631A Triple Output DC Power Supply

Software:

- Arduino IDE
- PJRC Teensy Software (Arduino IDE Extension) [1]
- IntuiLink Data Capture

## 3 Methods and Results

### 3.1 Control Input Design and Testing

#### 3.1.1 Glove Controller Design

First we worked on the glove controller, which the user will use to control the movement of the robot. The glove is made up of two flex sensors, an accelerometer, a muscle sensor, a push button, and a slide potentiometer. The flex sensors are attached to the index and middle fingers of the glove. The user chooses the forward or backward direction based on which finger is bent. To make the robot go forward, the user bends their index finger and to make the robot go backward, the user bends their middle finger. The accelerometer chooses the turning direction of the robot. If the user tilts their hand to the left, the robot turns left, and if the user tilts their hand right, the robot turns right. The layout of the glove with the two tilt sensors and accelerometer is shown below.

In addition to the sensors shown in Figure 1, the user wears an EMG muscle sensor which detects the level of muscle activity. The user controls the speed of the robot with this sensor. The more the user flexes their muscle, the faster the robot goes. Hence, the direction is chosen with the sensors pictured in Figure 1, and the speed is chosen using the muscle sensor. The muscle sensor is connected to the user via electrode sensor pads. The proper hookup of these pads is shown in Figure 2 below.

As can be seen in Figure 2 above, the electrode pads should be connected with the red

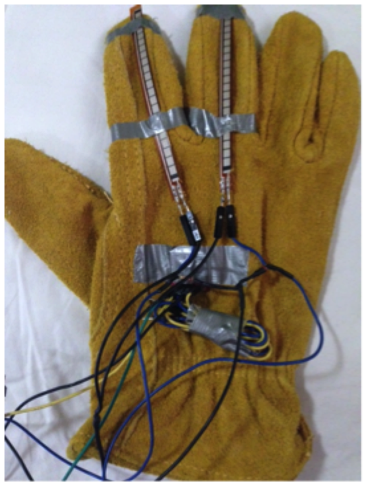


*Figure 1: Glove with tilt sensors and accelerometer*

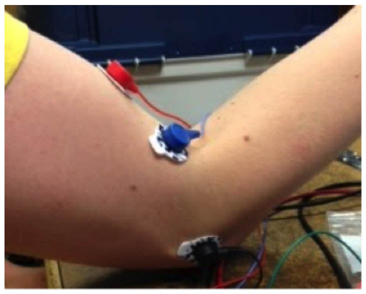


*Figure 2: Proper connection of the electrode sensor pads to the muscle*

connector on the middle of the muscle, the blue connector on the bottom of the muscle, and the black connector on a bone near the muscle. Proper connection of the sensor pads is very important to get a clear signal from the sensor. The muscle sensor provides a very interactive way to control the speed of the sensor. One drawback with EMG control is that the electrode sensor pads are supposed to be used only once and then thrown away, so it is not very feasible to pass the muscle controls between users. Due to this fact, we designed a second control mode which uses a different type of speed control. The user can enter the second mode by pressing a push button. In the second mode, the user still uses the glove pictured in Figure 1 to choose the direction of the robot, as the glove is easy to switch between multiple users, but the speed is then chosen via the slide potentiometer. There is also a LCD screen which prints important sensor values and tells the user which mode they are in for convenience.

### 3.1.2 Muscle Sensor Testing

We tested the EMG muscle sensor to see how well it responds to a user's muscle activity. We powered the muscle sensor with the power supply and connected the output signal to the oscilloscope, as shown in Figure 3 below.

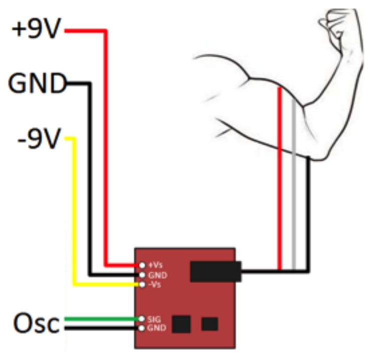


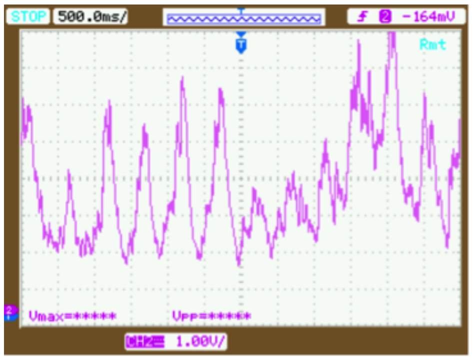
*Figure 3: Muscle sensor testing connections*

We did testing with +-9V, as that is the recommended voltage for the sensor. We found that the sensor responds very well to muscle activity. Plots of the signal output for the resting position, high bursts of muscle activity, and gradual increase in muscle activity are shown below.

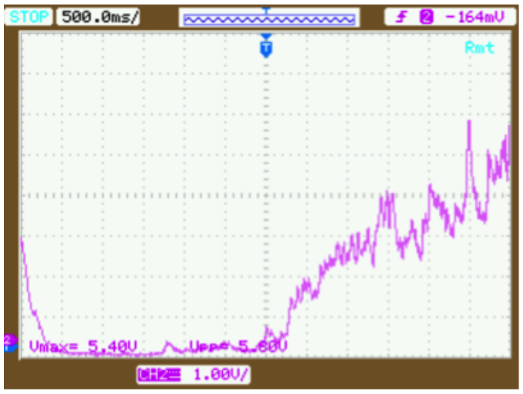
As can be seen in Figure 4 above, the muscle sensor detects muscle activity very well. When the user is resting, the output signal is very steady at a low voltage. When the user makes drastic movements, the output signal has huge spikes in voltage. It is also possible for the user to gradually increase the signal voltage by gradually flexing harder. We were able to get a good range of low, medium, and high amount of muscle activity that the user can easily differentiate between and maintain comfortably.

Although ±9V is the recommended input voltage, it is too high for the Teensy. We then powered the signal with +3.3V from the Teensy and -5V from a battery pack. Although the positive and negative voltage rails were not equal, the hardware had no problem working correctly. The positive rail was powered by the Teensy so that it did not receive input signals that would damage it. We then connected the output signal

(a)



(b)



(c)

*Figure 4: Muscle sensor output signal: (a) no muscle activity/resting; (b) high spikes of movement; (c) gradual increase in muscle activity*

of the muscle sensor to an analog pin on the Teensy in order to read the values that the sensor produces. We found that the sensor produces a very steady signal for the different ranges of muscle activity. There is a significant difference in the output value based on the muscle activity of the user. We tested the sensor for various ranges of muscle activity. The results are seen below in Table 1.

| Muscle Activity | Sensor Value |
|---|---|
| Resting | 85-95 |
| Low | 100-150 |
| Medium | 180-250 |
| High | 400-1023 |

*Table 1: Sensor values for various muscle activity ranges*

As shown in Table 1 above, we were able to get good ranges for the different muscle activity levels. They are easy to achieve and maintain by the user. The four ranges shown above correspond to the different robot speeds, where resting is our lowest speed and high is the maximum speed.

### 3.1.3 Flex Sensor and Accelerometer Testing

The accelerometer has three axes on it; however, we only use one axis, as we only need to be able to detect tilting in one direction. Therefore, we leave two of the axis signal outputs unconnected. We tested the sensor with it connected to the Teensy to see what values are outputted for different tilt ranges.

| **Tilt** | **Sensor Value** |
|---|---|
| Left | $V \geq 670$ |
| Right | $V \leq 550$ |

*Table 2: Sensor values for left and right tilt orientation*

As seen above in Table 2, when the user tilts the glove left, the value read from the accelerometer is above 670, and when the user tilts the glove right, the value read is below 550. When the sensor is outputting a value between the two values seen in Table 2 it constitutes as the user keeping their hand flat, and therefore the robot does not turn.

The flex sensors were tested in a similar manner. The flex sensor is a variable resistor, and we connected each of them in series with a 22k$\Omega$ resistor, creating a voltage divider, where the upper resistor was the flex sensor. The center node was then connected to an Analog Input pin on the Teensy.

We found cutoff values for when the flex sensor is bent an appropriate amount. The value is 550 for the forward control flex sensor and 600 for the backward control flex sensor. It is interesting to note that the values for the two flex sensors are slightly different. This is most likely due to inherent tolerances within the flex sensors or resistors.

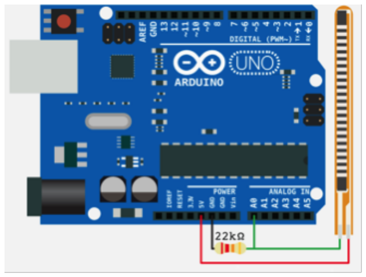


*Figure 5: Connection of a flex sensor to an Arduino (in our case a Teensy)*

#### 3.1.4 Push Button and Potentiometer Implementation

As stated earlier, the push button and slide potentiometer are used to implement a second mode during which the user can control the speed of the robot with the potentiometer rather than the EMG muscle sensor. This second mode is entered when the push button is pressed. Then the user can return back to the first mode by pressing the button again. The potentiometer controls the speed from 0% to 100% based on what position it is slid to.

## 3.2 Wireless Connection and Final Implementation

#### 3.2.1 Wireless Setup and Code

To communicate wirelessly with the robot, we chose to use a pair of ESP8266 wireless modules, one of which created a WiFi Access Point and ran the TCP server, the other of which connected to the AP. Although commands were only sent to the robot, bi-directional communication is feasible with this chip. A Teensy 3.1 and a Teensy LC were chosen due to having hardware serial with hardware buffers and 3.3V logic like the ESP8266 module, making communication easier than with an Arduino. We already had a Teensy 3.1 for the transmitting end and the LC version was purchased for the receiving end. This configuration led to an implementation using three microcontrollers to translate an array of inputs into outputs, as shown in the block diagram in Figure 7. The connection was handled using bi-directional logic level converters using the

MOSFETs provided in our lab kit.

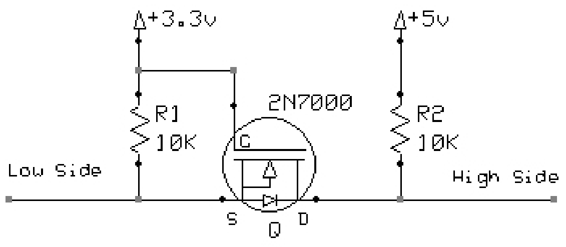


*Figure 6: Example bi-directional logic level shifter schematic*

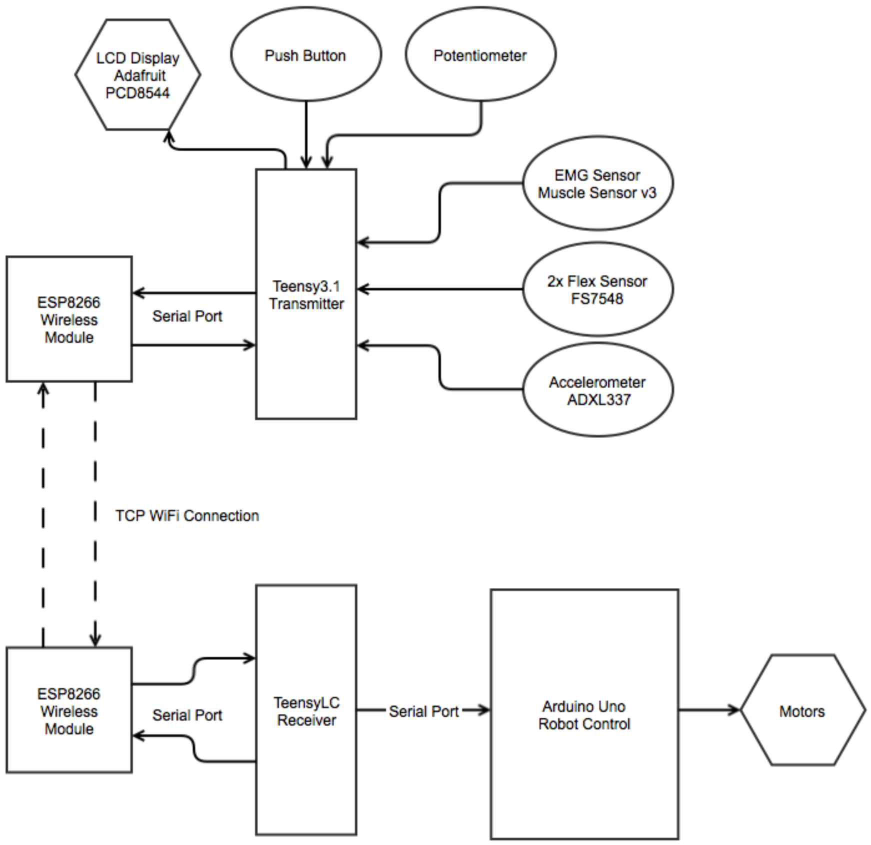


*Figure 7: Block diagram*

Below is a brief description of the purpose of each microcontroller, and pseudocode of the software running on each.

**Transmitter TeensyLC:** Configure WiFi AP, collect sensor data, transmit commands

Setup routine:

- Initialize inputs and Adafruit display
- Start serial communications
- Attach interrupt to push button handler function
- Configure ESP8266 as access point

Main loop:

- Check which sensor is activated and in which way
- Prints values to LCD screen
- Send a forward, backward, left, right, or stop command in response to glove controller sensors
- Send a speed command in response to EMG or potentiometer readings

**Receiver TeensyLC:** Connect to WiFi AP, translate and pass on commands

Setup routine:

- ESP connection and configuration
- Start serial communications

Main loop:

- Detect which direction and speed is being sent by the transmitter
- Communicate with the Arduino on the robot (wired)

**Robot Control – Arduino Uno:** Execute robot commands

Setup routine:

- Start serial communications

Main loop:

- Position control code from previous labs
- Call different direction and speed functions based on what command is received

The code is several pages long, so we have created a zip folder containing the final code, which can be found in the references. The code for the transmitter/sensor input TeensyLC is located in ECEN_TX. This microcontroller maintains configuration of the wireless module, reads sensor input, computes commands for the robot, and sends these to the wireless transmitter module.

The code for the receiver TeensyLC is located in ECEN_RX. This microcontroller connects to the wireless receiver module, reads commands from it, and passes these commands to the Arduino.

The code for the robot control Arduino Uno is located in Robocode. This microcontroller reads commands from the wireless receiver module and translates them into actions (in this case, driving motors).

#### 3.2.2 Packaging

Several of the input devices, including the accelerometer, the flex sensors, and the EMG were connected to a glove. We also designed a small box that would be attached to the user's arm. The Teensy, an antenna, batteries, and a screen that displayed the output readings from the glove were placed in the box. The second receiver Teensy was placed on the robot next to all the circuitry from the previous labs. The information from the receiver Teensy was then fed into the Arduino on the robot, which led to motor control.

## 4 Further Explorations

If this project were extended and improved upon, a good course of action would be to spend more time on the packaging. For our hand controller we used a garden glove, as that was the best option with the amount of time that was given. However, if there was more time, it would have been a great idea to design and manufacture a glove with plastic coating that made it look like a robot hand. This would have made the project look more complete at the end and would probably result in a more entertaining experience for the user. In addition, more care should have been taken when connecting the EMG muscle sensor up to the batteries. The sensor has both a positive and a negative input voltage, so the user should be very careful when connecting the batteries to ensure that the positive pin receives a positive voltage and the negative pin receives a negative voltage. If the negative pin receives any amount of positive voltage, the sensor will heat up and break.

## 5 Conclusion

Our group successfully demonstrated wireless communication and motor control with the glove. We were able to implement robot direction control with the flex sensors and accelerometer as well as speed control with both the EMG muscle sensor and the slide potentiometer. The wireless communication proved to be the most difficult part of the project. We initially struggled with establishing the connection between the transmitter and receiver. There was also a lot of code to go through and a few bugs in the program that caused information to be lost between the components. It was more difficult than the last wireless system, but it did have its benefits. WiFi increased our range for the distance the robot could get from the operator while still responding. Using bluetooth could have been an ulterior method that might have been easier to build.

## 6 References

[1]. Teensy Arduino IDE Extension: `https://www.pjrc.com/teensy/td_download.html`

[2]. Teensy Reference: `https://www.pjrc.com/teensy/`

[3]. TeensyLC Product Page: `https://www.pjrc.com/teensy/teensyLC.html`

[4]. ESP8266 Datasheet: `https://nurdspace.nl/images/e/e0/ESP8266_Specifications_English.pdf`

[5]. ESP8266 Reference: `https://nurdspace.nl/ESP8266`

[6]. GitHub Software Repository: `http://github.com/Dacilndak/2270-g121`

[7]. GitHub Software Archive: `https://github.com/Dacilndak/2270-g121/archive/master.zip`