Electromyography Car

Design Documentation

Sam Whiting
Sara Larson
December 2015

Table of Contents

1. Introduction

This document describes the design of the electromyography-controlled car.

2. Scope

This document discusses the hardware and software design used to create the EMG (electromyography) controlled car It does not provide details about the mechanical design, such as circuit board layout or button and switch construction. The hardware schematics are included in appendix A. The complete code, written in C, is included in appendix C.

3. Design Overview

The following section outlines the general operating principles and requirements for the EMG car.

3.1 Requirements

The following are requirements for the Electromyography car project.

1. The EMG signal from a user shall be read using electrodes placed on certain muscle groups.

2. The EMG signal from a user shall be read, amplified, and rectified using the Advancer Technologies muscle sensor v3 board.

3. EMG signal shall be interpreted by the Tiva C series TM4C123GH6PM microcontroller.

4. The microcontroller shall sample the EMG signal at a rate of 500 Hz (every 2ms)

5. The remote-controlled car shall be controlled by the microcontroller, and shall drive forward upon the flexing of certain muscle groups.

6. The remote-controlled car shall stop motion within 1 second of the user relaxing the muscle group beneath the threshold.

7. The noise from the remote-control device shall be reasonable dealt with, so as to not interfere with the EMG signal received by the muscle sensor board.

8. The code driving the microcontroller shall be written in the C programming language.

9. The user shall have options to switch directions to either forward and backwards and left or right through two switch modules located on the breadboard.

3.2 Dependencies

The dependencies are required for the Electromyography car project.

1. A Tiva C series TM4C123GH6PM Microcontroller
2. A muscle sensor v3 board by advancer technologies
3. A +- 9v power supply
4. A 5v power supply
5. A 16MHz crystal oscillator
6. Shielding material to reduce noise

3.3 Theory of Operation

The following section provides a simple overview of the system operations. For a top-level diagram, see figure 3.3.1.

A positive and negative diode are placed over the same muscle group, in close proximity. A third, grounding diode is placed at the end of the muscle group, over a boney region. The signal read from the diodes is sent to the muscle sensor board.

The muscle sensor board rectifies and amplifies the EMG signal input. The microcontroller interprets the input, and if a certain threshold is reached, (req. 6) the radio command will be sent in order to move the car forward. If the threshold is not reached, the car will receive no command, resulting in the end of movement by the car.

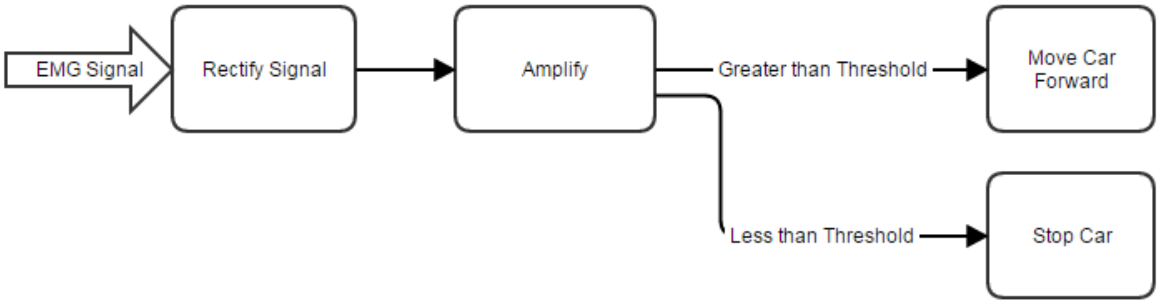


Figure 3.3.1: Top Level Diagram

4. Design Details

This section covers the software and hardware design in greater depth.

4.1 Hardware Overview

This section explains the hardware design, along with appendix A, which contains a detailed hardware schematic. Figure 4.1.1 provides a simplified overview of the hardware layout.

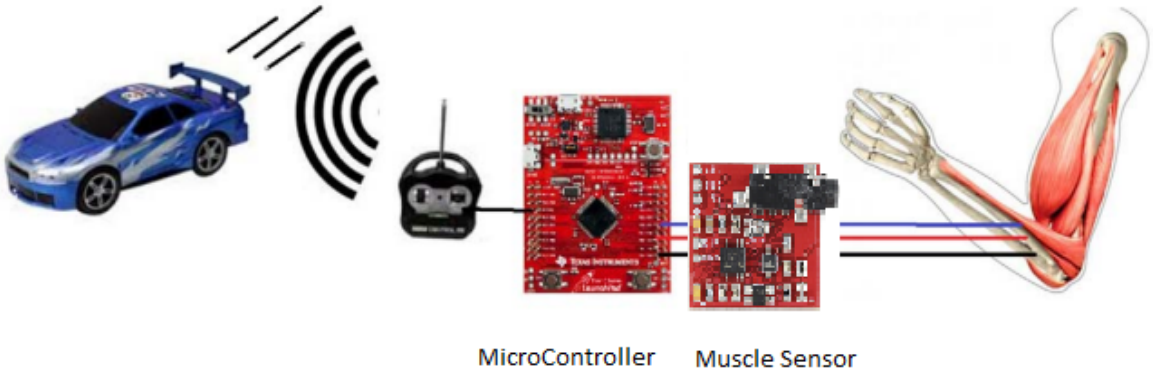


Figure 4.1.1: Basic Hardware Layout

4.1.1 Components

The following components were used for the EMG car project.

Two Toshiba TLP227G photocoupler (photo relays)
Two 4-position DIP switches
One 220 Ohm resistor
One Muscle Sensor V3 board (advancer tech)
One generic R/C car
One Tiva C microcontroller, with power supply and 40MHz oscillator

4.2 Input/Output Overview

This section explains the various inputs and outputs of the system.

4.2.1 MicroController Outputs

Pin A5 on the microcontroller controls the output for the EMG car. When driven high, the pin sources current through the 220 Ohm resistor, and across both photorelays. While high, the photo relays allow current flow on the opposite side. The connections on the other side are directly wired to the radio transmitter across the DIP switches. When the output pins on the photorelay allow current flow, the corresponding function on the transmitter is enabled. By using the DIP switches to control which functions are in circuit, it is possible to drive the car forward, backward, left, and right. See section 4.4 for more details about the system controls.

While low, the photo relays remain in their open configuration, not allowing current to flow on the output side. Pin A5 has a pull down resistor, resulting in this open configuration by default. The pin is only driven high when the analog input reaches a certain threshold. For this project, that threshold has been set at ADC code 0x200. See figure 4.2.1 for a diagram of the output functionality, as a result of the input. Additionally, see section 4.2.2 for a description of the analog input.

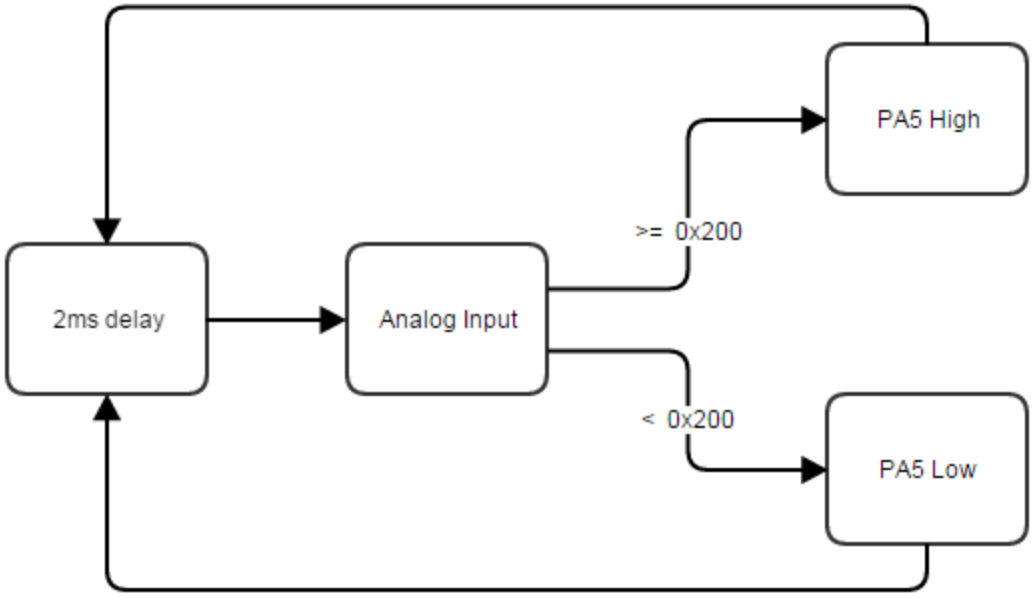


figure 4.2.1 - microcontroller Input/Output flow

4.2.2 MicroController Inputs

Pin D0 on the microcontroller is the analog input. This input comes from the muscle sensor analog output (see section 4.2.1). The ADC samples the voltage every 2 ms, and uses that value to toggle the output states. The output states as a result of the input can be seen in figure 4.2.1. See section 4.2.3 for a more detailed description of the muscle sensor signal.

Appendix B, figures A5, A6, and A7 show the EMG input signal, and the resulting output on pin A5. EMG pulses of different frequencies are shown in order to demonstrate the response of the system.

The microcontroller takes a 5V input. The schematic and type of power regulator (outputs 3.3V) can be seen in appendix A.

4.2.3 Muscle Sensor Outputs

The analog output on the muscle sensor board is an analog signal resulting from the diode inputs (see section 4.2.4). The output signal is filtered, rectified, smoothed, and amplified with an approximate gain of 10,000. A schematic of the muscle sensor board has been provided by advancer tech, and can be found in appendix A.

Appendix B, figures A1, A2, and A3 show the EMG signal output from the muscle sensor board.

4.2.4 Muscle Sensor Inputs

The inputs for the muscle sensor come from three diodes that are attached to a muscle group on the user. The positive and negative (red and blue) diode wires are connected over the body of the muscle group, in close proximity to each other. The ground diode (black) is connected to a bony region near the muscle group, but not over any muscles.

The three diode wires feed into a 3.5mm port on the muscle sensor board, leading to the on-board op-amps. See section 4.2.3 for more information on the handling of the input signal before it is output.

The muscle sensor board takes +9V and -9V inputs in order to rectify and amplify the output signal.

4.3 Noise Handling

The amplifying effect of the muscle sensor board results in a serious noise issue. Any noise that is combined with the diode inputs is heavily amplified, resulting in a signal that is unusable. See Appendix B, figure A4 for the noise-trapped signal.

Some noise results from sharing grounds between devices. Optocouplers (photo relays) were used in order to electrically isolate the systems. In this way, any common ground noise was attenuated.

The most significant source of noise is the radio transmitter. When sourcing a radio signal, the controller broadcasts loudly, compared to the EMG signal. This radio noise is picked up by the diode wires, which act as antennas. Since the diode wires feed into the muscle sensor input, this noise is amplified, resulting in a noise loop, as seen in figure A4. By shielding the wire using tinfoil and plastic, the radio noise is sufficiently attenuated, and a readable signal is output.

Figures A1, A2, and A3 show the EMG signals in a noise-free environment. A4 shows the noise-loop as a result of the radio signals. Figures A5, A6, and A7 show the shielded and isolated signal, as well as the corresponding output signal as a result of reaching the analog threshold.

4.4 Controls

The following section provides an explanation of the car controls, using the DIP switches. For information on the control using the EMG input, see section 4.2.

The DIP switches are used in place of multiple EMG inputs. Since the cost of an EMG input for this system is about $60, DIP switches are used instead to demonstrate functionality of multiple EMG inputs at a lower cost. By opening and closing the switches in certain configurations, it is possible to drive the car forward, backward, left, and right, and in combinations along each axis. (front-right, back-right, front-left, back-left). For the switch configurations, see figure 4.4.1.

	DIP Switch Configuration (left to right)							
	1	2	3	4	5	6	7	8
forward	closed	closed	open	open	open	open	open	open
back	open	open	closed	closed	open	open	open	open
left	open	open	open	open	closed	closed	open	open
right	open	open	open	open	open	open	closed	closed

figure 4.4.1

4.5 Initializations

Port D pin D0 is set as the alternate function analog input. Timer 0 is initialized and set to 2ms periodic mode. Timer 1 is initialized and set to 500ms periodic mode. Port A pin 5 is initialized as a digital output. Timer 2 is initialized to 10ms periodic mode. The ADC is initialized on AIN7 pin 61 I Analog Analog-to-Digital converter input 7. The phase locked loop is set to provide a 40MHz clock signal.

5. Testing

Following are the various tests used to verify the requirements listed in section 3.1.

5.1 Hardware Test

This test verified that the car was communicating throughout all of the hardware and was starting and stopping within the requirements located in 3.1.5 and 3.1.6. This test also verified that the two 4-bit dip switches functioned as expected by allowing the user to switch from the forward direction to the backwards direction as well as allowing the user to add direction either left or right as outlined in section 3.1.9.

5.2 EMG Signal Test

This test is verified by connecting the EMG pins to the oscilloscope and confirming that a signal appears that matches the muscle pulses outputted. Next, the EMG pins are connected to the muscle sensor V3 board and we confirm that we still see the same signal without noise. This was confirmed through images taken while the EMG pins were connected to the logic analyzer in Appendix B.

5.3 Functional Test

This test is verified by powering the system, and allowing a user to control the car by flexing the muscle group that the diodes are connected to. Observation of the car movement confirms that the system is behaving as expected, and responding to the muscle impulses.

5.4 Noise Test

This test is verified by using the logic analyzer and oscilloscope to measure the EMG signal for noise. The amplitude of the EMG signal must be significantly larger than that of the noise, in order to result in a readable signal. This test can be ran while diodes are connected to a user, in order to source the EMG signal. Results of these tests can be found in appendix B.

6. Conclusion

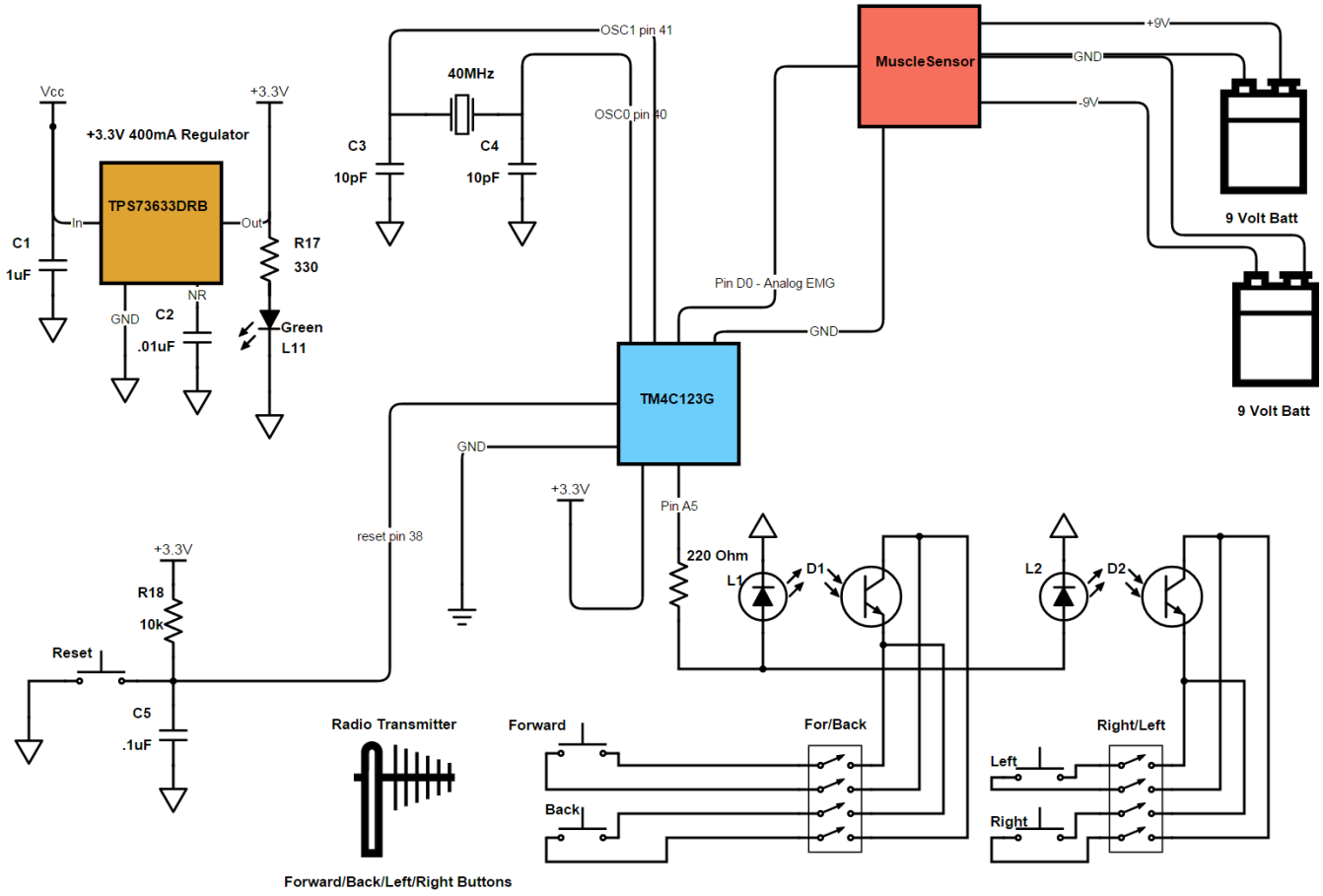
This design produced a working EMG car that operated according to the requirements. All the tests in section 5 verified that the project functioned as required.

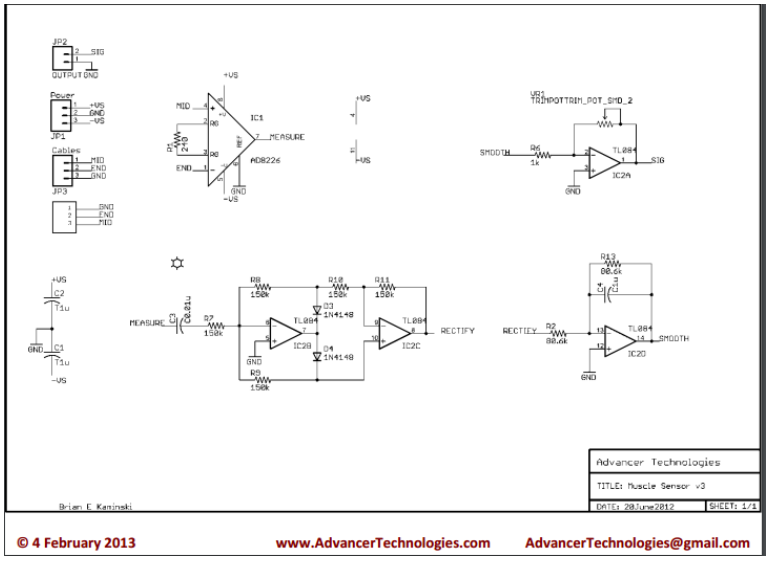
The DIP switch controls functioned as required, but did not provide an optimal level of control. By using more EMG input channels, the car could respond more naturally to a user's intentions.

The hardware used to analyze the signal was more than capable of completing its simple task, and a smaller or cheaper microcontroller could have been used for this project. The addition of multiple EMG inputs would have used the microcontroller capabilities more fully, if implemented.

The code used to run this system could have been completed using fewer lines of code. The methods we used to develop the system code was focused on making it easy to understand and modify. This comes at the cost of speed and length, but overall was worth the trade-off, as it allows for adding modification or additional functionality easily.

Appendix A: Schematic





Appendix B: Logic Analyzer Images

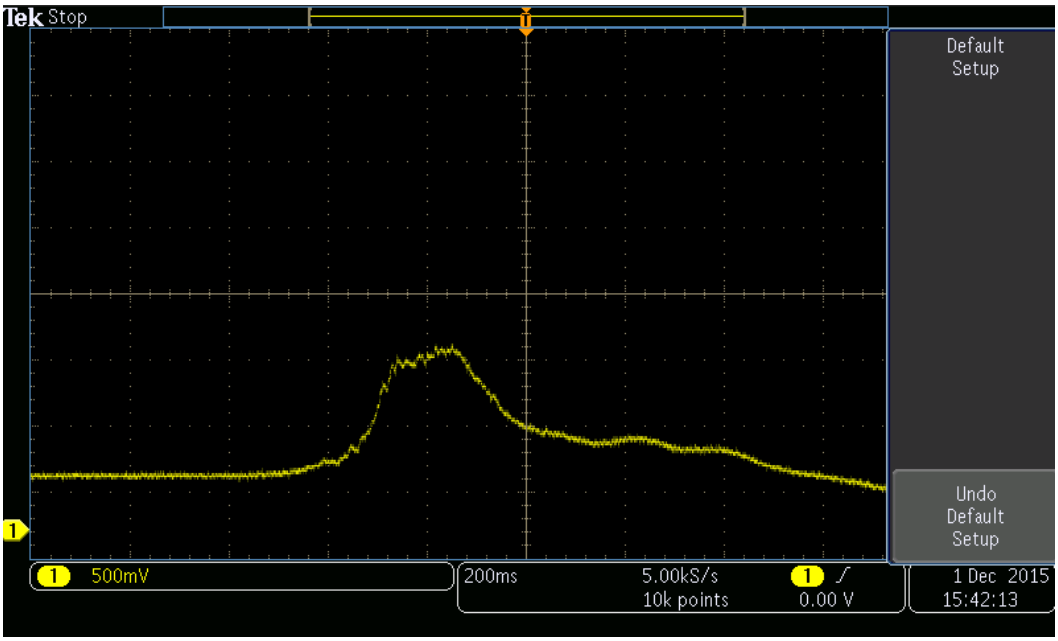


Figure A1

The above image shows output from the muscleSensor v3 board going to the ADC of the Tiva C. The ~1 volt pulse shown was captured when a muscle was flexed, creating this electromyography signal.

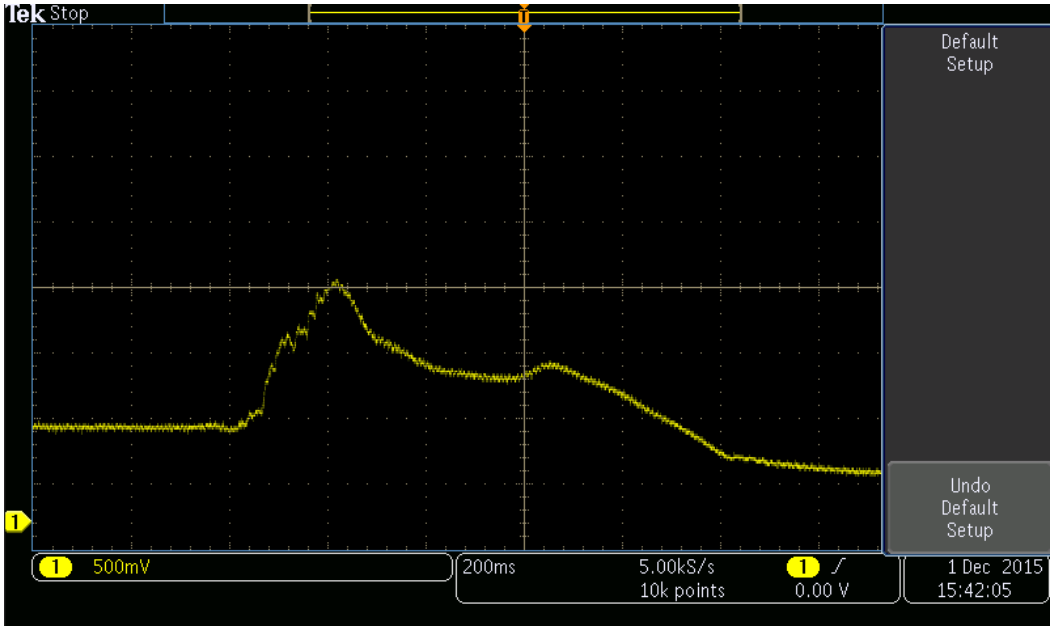


Figure A2

The above image shows another output from the muscleSensor v3 board going to the ADC of the Tiva C. This signal is similar to the one above, but the user flexed a second, smaller time after the initial pulse.

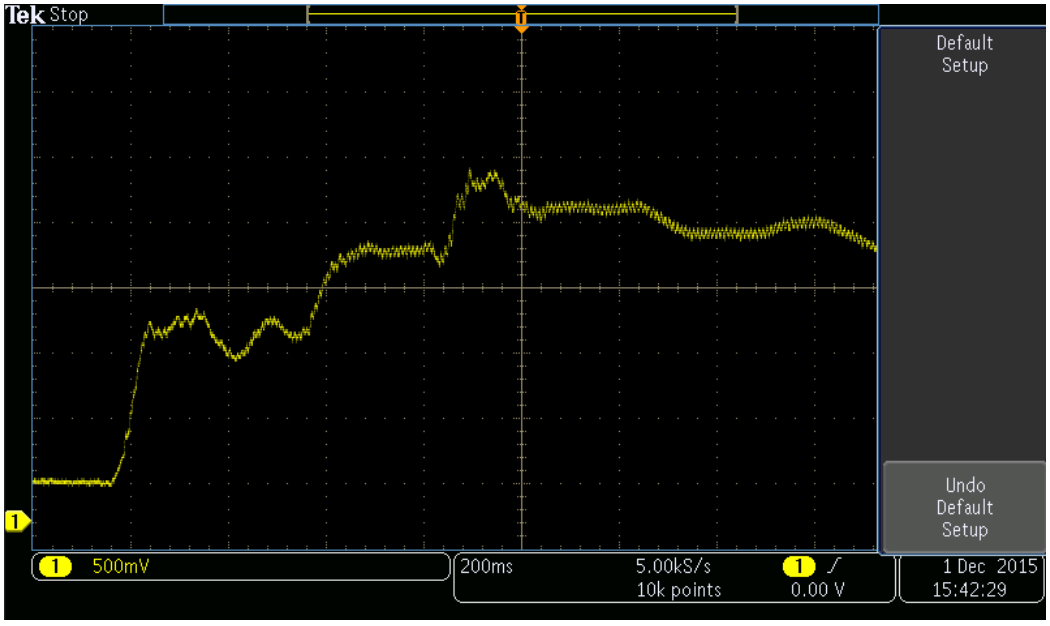


Figure A3

The above image shows another output from the muscleSensor v3 board going to the ADC of the Tiva C. This signal is a result of the user flexing without restraint, resulting a signal that steps up and maintains a high relative voltage.

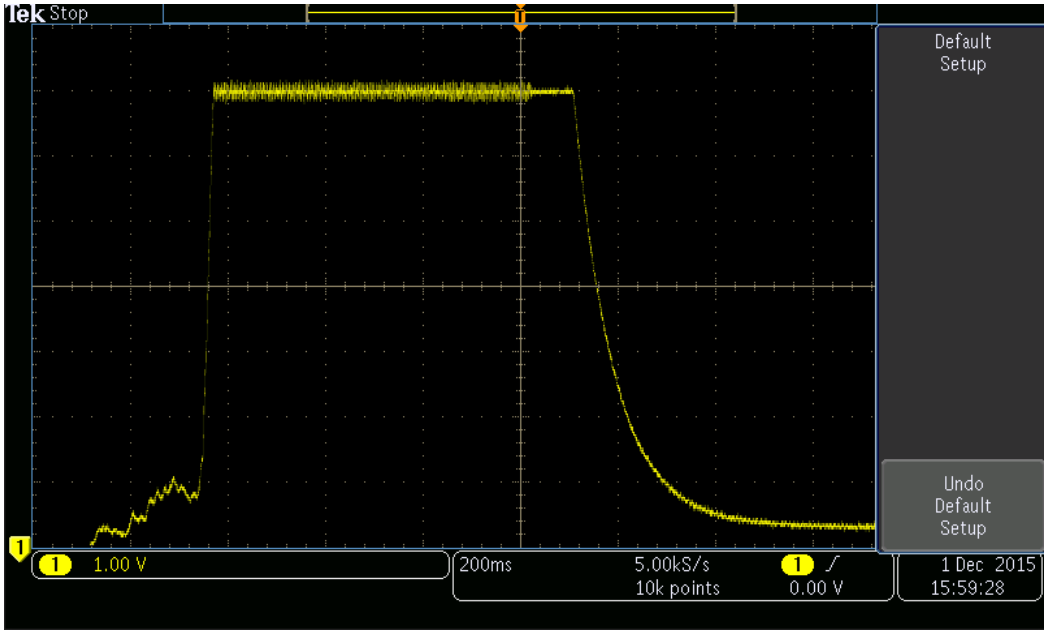


Figure A4

The above image shows the unshielded output from the muscleSensor v3 going to the ADC of the Tiva C. This output reaches its maximum, and is sustained, due to noise from the radio controller, until the controller device is disconnected. This disconnect results in the signal falling off rapidly.

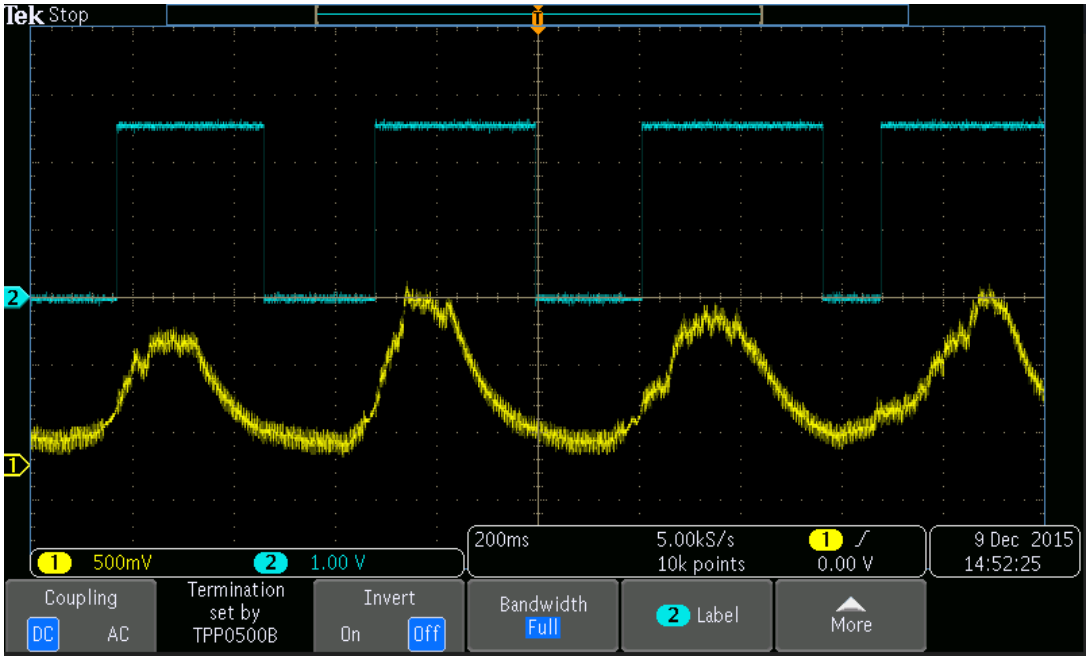


Figure A5

The above image shows the shielded output from the muscleSensor v3 going to the ADC of the Tiva C in yellow. Also shown is the optocoupler activation, driving the signal that turns the motor on. (shown in blue)

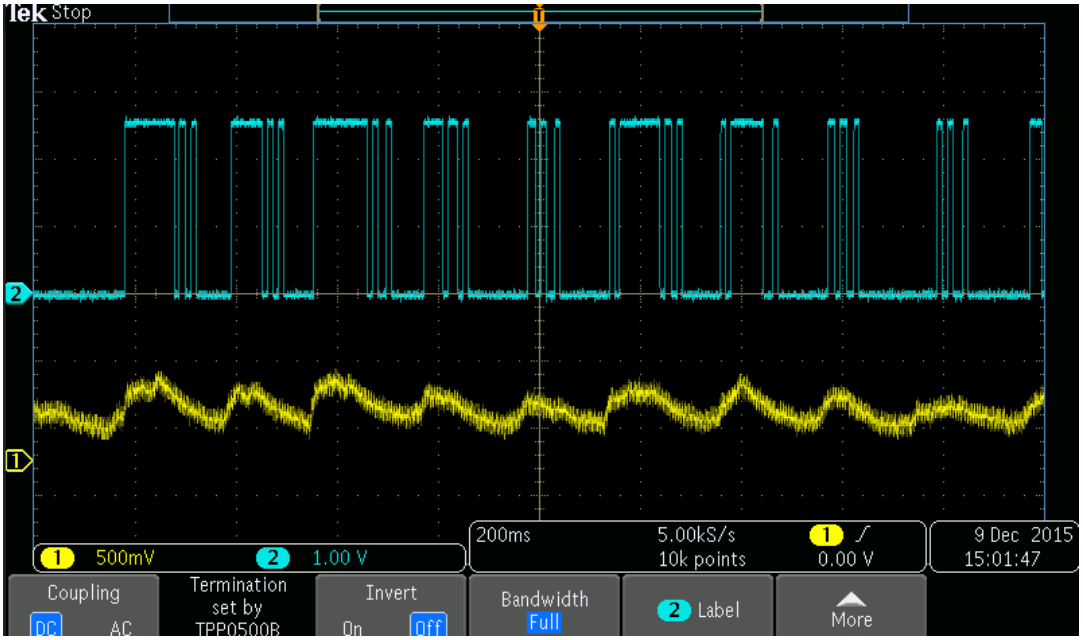


Figure A6

The above image shows rapid, shielded EMG pulses (yellow) and the resulting optocoupler activation, driving the motor signal (blue).

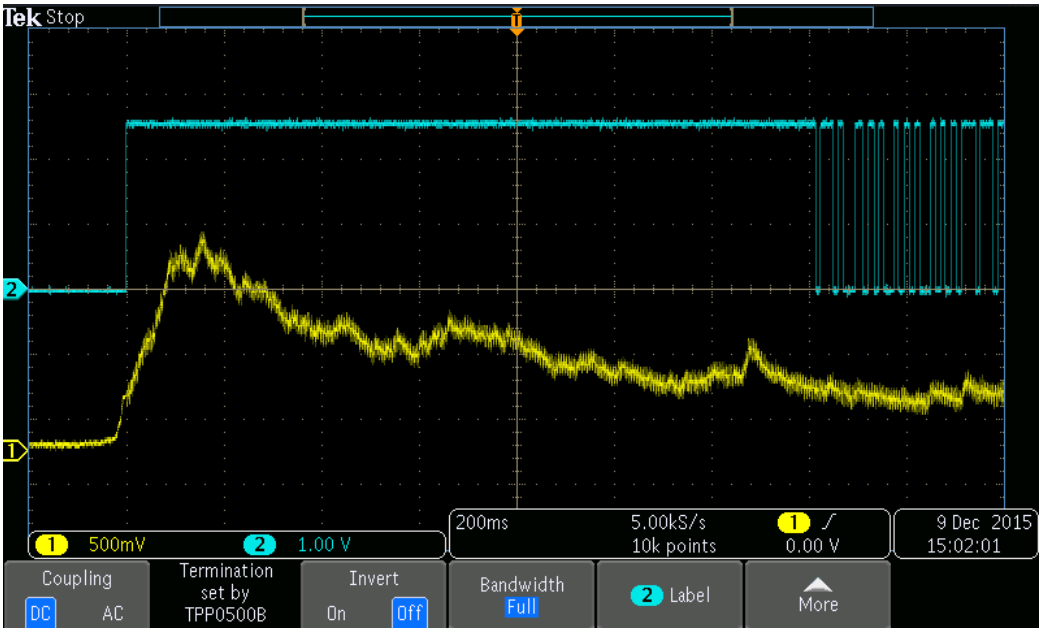


Figure A7

The above image is another EMG pulse (yellow), sustained in this case, resulting in a steady ‘on’ signal for the motor signal (blue).

Appendix C: Code

```
#include "tm4c123gh6pm.h"

//globals
short voltage;

//*******initializations*******

//PLL to 40MHz - 0x0240.1540 to RCC
void setPLL(void){
    SYSCTL_RCC_R = 0x02401540;
}

//Pin D0 as alt function analog input
void initPortD(void){
    SYSCTL_RCGCGPIO_R |= 8;
    GPIO_PORTD_AFSEL_R = 1;
    GPIO_PORTD_DEN_R = 0;
    GPIO_PORTD_AMSEL_R = 1;
}

//set timer 0 to 2ms periodic
void initTimer0(void){
    SYSCTL_RCGCTIMER_R |= 1;    //General-Purpose Timer 0 clock
    TIMER0_CTL_R &= 0xFFFE;          //disable timer 0
    TIMER0_CFG_R = 0;                              //32 bit mode
    TIMER0_TAMR_R = 2;                      //periodic mode
    TIMER0_TAILR_R = 80000;          //start value: 40MHz * 2ms = 80000
    TIMER0_CTL_R |= 0x20;            //send output to ADC
    TIMER0_CTL_R |= 1;                      //enable gptm 0
}

//set timer 1 to 500ms periodic
void initTimer1(void){
    SYSCTL_RCGCTIMER_R |= 2;    //General-Purpose Timer 1 clock
    TIMER1_CTL_R &= 0xFFFE;          //disable timer 1
    TIMER1_CFG_R = 0;                              //32 bit mode
    TIMER1_TAMR_R = 2;                      //periodic mode
    TIMER1_TAILR_R = 20000000;//start value: 40MHz * 500ms = 20,000,000
    TIMER1_CTL_R |= 1;                      //enable gptm 1
}
//Pin A5 Digital Output
void initPortA(void){
    SYSCTL_RCGCGPIO_R |= 1;
    GPIO_PORTA_PDR_R = 0x20; //default low
    GPIO_PORTA_DIR_R = 0x20;
    GPIO_PORTA_DEN_R =0x20;
}
```

```
//set timer 2 to 10ms periodic
void initTimer2(void){
    SYSCTL_RCGCTIMER_R |= 4;    //General-Purpose Timer 2 clock
    TIMER2_CTL_R &= 0xFFFE;          //disable timer 2
    TIMER2_CFG_R = 0;                //32 bit mode
    TIMER2_TAMR_R = 2;               //periodic mode
    TIMER2_TAILR_R = 400000;    //first start value: 40MHz * 10ms = 400,000
    TIMER2_CTL_R |= 1;               //enable gptm 2
}



//AIN7 61 I Analog Analog-to-digital converter input 7.
void initADC(void){
    SYSCTL_RCGCADC_R = 1;    //clock for adc0
    initPortD();                              //pd0 as analog input
    ADC0_ACTSS_R = 0;                //disable sequencers
    ADC0_EMUX_R = 0x5;        //use timer for sampler 0
    initTimer0();
        //default 125kSamples/sec
        //sequencer 0 has highest priority
    ADC0_SSMUX0_R = 0x7;    //first sample is from ain 7 (pd0)
    ADC0_SSCTL0_R = 6;        //first sample has interrupt, and is the last sample
    ADC0_IM_R     = 1;                        //sequencer 0 has interrupt
    ADC0_CC_R = 0;                   //use PLL main osc
    ADC0_ACTSS_R |= 1;        //enable sequencer0
    NVIC_EN0_R |= 0x4000;    //enable interrupt 14 at system level
}


//*******interrupt handlers*******

//adc interrupt
void ADC0SS0_Handler(void){
    ADC0_ISC_R |= 1;          //clear interrupt
    TIMER0_ICR_R = 1;         //reset timer 0
    voltage = ADC0_SSFIFO0_R;    //copy in data from sequencer 0;
}


//*******functions*******

//500ms Delay
void Timer_500(void){
    TIMER1_ICR_R = 1;           //reset timer 1
    TIMER1_TAV_R = 20000000;
    while((TIMER1_RIS_R && 1) ==0);
}
```

```
//10ms Delay
void Timer_10(void){
      TIMER2_ICR_R = 1;          //reset timer 2
        TIMER2_TAV_R = 400000;
        while((TIMER2_RIS_R && 1) ==0);
}

int main(void){
  setPLL();
  initADC();
  initTimer1();
    initPortA();
  initTimer2();


    voltage = 0;
    GPIO_PORTA_DATA_R=0x0;

    while(1){
        while(voltage < 0x200); //does nothing while voltage is less than 200
        GPIO_PORTA_DATA_R = 0x20; //turn optocoupler on
        //Timer_10();
        while(voltage > 0x200);
        GPIO_PORTA_DATA_R = 0x0;
        //Timer_10();

    }

}
```