Lab 3 - Waldo

Sheil Sarda sheils@seas.upenn.edu

3.1 Waldo Input

3.1.1 Build Plan and Fabrication

Dims of dog for input

Length	8 inches
Width	6.5 inches
Potentiometer - fixed part	Radius: 0.125 inches
Potentiometer - rotating	Radius: 0.062 inches

Notes from check-off with Walter 2/19

- Since the servos we have to use for the actuation part of the assignment are quite tiny, may need to shrink the design for that part
- Best way to account for Kerf is to use the Shell command with `shell outward` enabled. Use shell radius of `0.007` inches

- Line width of DWG files for lasercutting should be `0.05` mm, as opposed to the inches unit mentioned in the slides

Electrical Components

Part	Quantity	In Kit
D-shaft potentiometers	2	Pick-up
Stem potentiometer (3310Y-001-103L-ND)	2	Have
SG90 Servo	2	Have

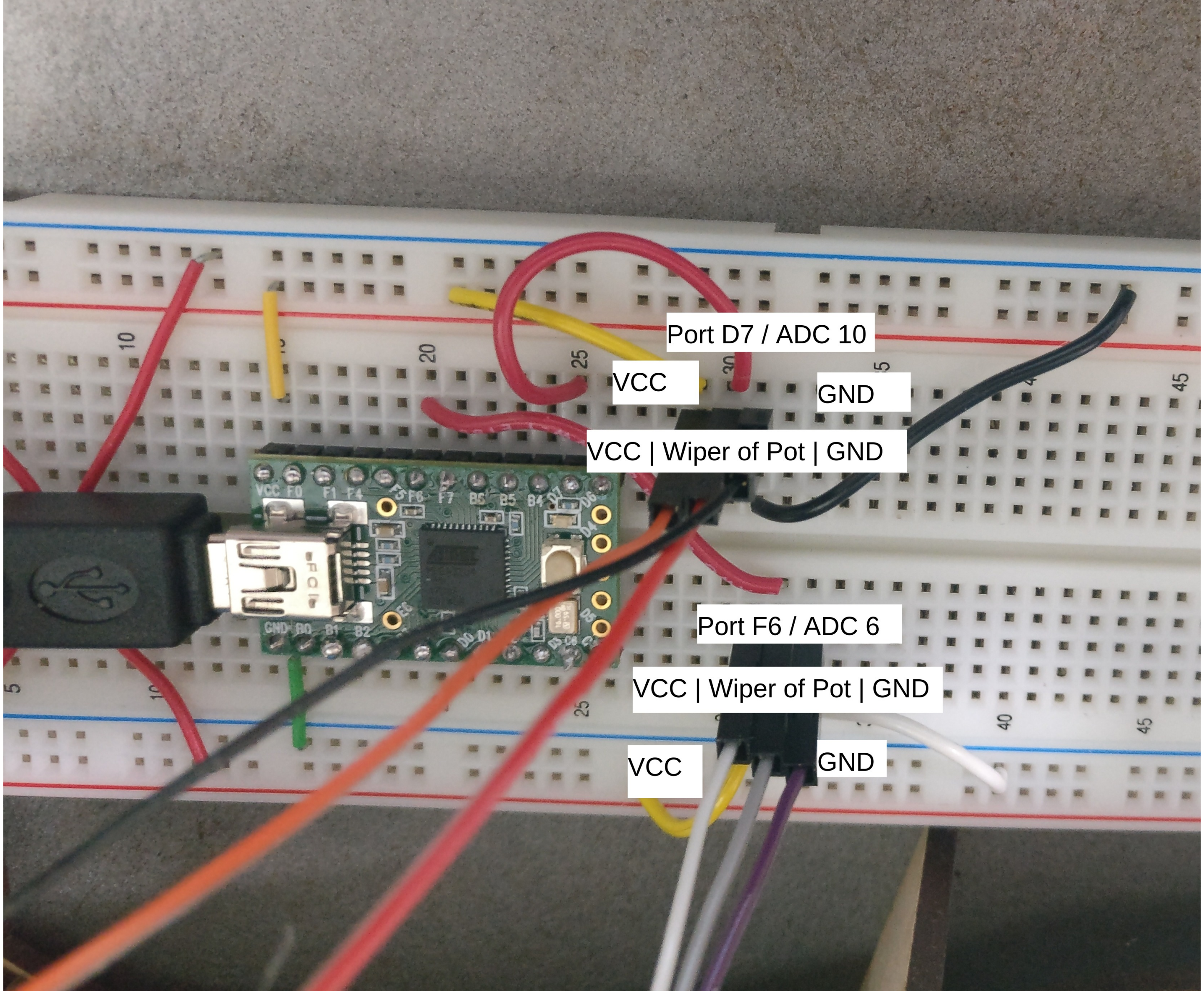
3.1.2 ADC Subroutines

```
#include "teensy_general.h"
#include "t_usb.h"
#include <stdbool.h>
#include <string.h>

#define CLOCK_SPEED 16e6
#define PRESCALAR 1024 // freq of 15.625kHz
#define TARGET 10e3
#define USB 1
#define MAX_ADC 1023
#define MAX_ANG 300

void setup_ADC(char adc_num){

    set(ADMUX, REFS0); // AVcc

    // 128 ADC prescalar
    set(ADCSRA, ADPS0);
    set(ADCSRA, ADPS1);
    set(ADCSRA, ADPS2);

    switch(adc_num){ // Disable digital input
        case 0 : set(DIDR0, ADC0D);   break;
        case 1 : set(DIDR0, ADC1D);   break;
        case 4 : set(DIDR0, ADC4D);   break;
        case 5 : set(DIDR0, ADC5D);   break;
        case 6 : set(DIDR0, ADC6D);   break;
        case 7 : set(DIDR0, ADC7D);   break;
        case 8 : set(DIDR2, ADC8D);   break;
        case 9 : set(DIDR2, ADC9D);   break;
        case 10: set(DIDR2, ADC10D);  break;
        case 11: set(DIDR2, ADC11D);  break;
        case 12: set(DIDR2, ADC12D);  break;
        case 13: set(DIDR2, ADC13D);  break;
    }

    set(ADCSRA, ADEN); // enable ADC
    set(ADCSRA, ADSC); // first conversion

}

void setup_next(char adc_num){

    unsigned int mask1 =
                        (1 << MUX0) |
                        (1 << MUX1) |
                        (1 << MUX2);

    ADMUX &= ~mask1;

    clear(ADCSRB, MUX5);

    switch(adc_num){
        case 0 :                        break;
        case 1 :    set(ADMUX, MUX0);   break;
        case 4 :    set(ADMUX, MUX2);   break;
        case 5 :    set(ADMUX, MUX2);
                    set(ADMUX, MUX0);   break;
        case 6 :    set(ADMUX, MUX2);
                    set(ADMUX, MUX1);   break;
        case 7 :    set(ADMUX, MUX0);
                    set(ADMUX, MUX1);
                    set(ADMUX, MUX2);   break;
        case 8 :    set(ADCSRB, MUX5);  break;
        case 9 :    set(ADCSRB, MUX5);
                    set(ADMUX, MUX0);   break;
        case 10:    set(ADCSRB, MUX5);
                    set(ADMUX, MUX1);   break;
        case 11:    set(ADCSRB, MUX5);
                    set(ADMUX, MUX1);
                    set(ADMUX, MUX0);   break;
        case 12:    set(ADCSRB, MUX5);
                    set(ADMUX, MUX2);   break;
        case 13:    set(ADCSRB, MUX5);
                    set(ADMUX, MUX2);
                    set(ADMUX, MUX0);   break;
    }

}

int read_adc(char next_adc){

    set(ADCSRA, ADSC); // first conversion

    setup_next(next_adc);

    while(!bit_is_set(ADCSRA, ADIF));

    unsigned int result = ADC;

    set(ADCSRA, ADIF); // clear compute flag
    return result;
```

```
}

int main(void){
    #ifdef USB
        m_usb_init();
        while(!m_usb_isconnected());
    #endif

    // set 1024 prescalar
    set(TCCR3B, CS32); set(TCCR3B, CS30);
    teensy_clockdivide(0); //set the clock speed

    setup_ADC(10);  // ADC10 or PD7
    setup_ADC(6);   // ADC6 or PF6

    unsigned int result1, result2;

    while(1){

        if(TCNT3 < TARGET * PRESCALAR / CLOCK_SPEED) continue;

        TCNT3 = 0;

        result2 = read_adc(10); // gives you ADC6
        result1 = read_adc(6);  // gives you ADC10

        result2 *= ((float) MAX_ANG / MAX_ADC);
        result1 *= ((float) MAX_ANG / MAX_ADC);

        #ifdef USB
            m_usb_tx_string("LEFT: ");
            m_usb_tx_uint(result2);
            m_usb_tx_string("  RIGHT: ");
            m_usb_tx_uint(result1);
            m_usb_tx_string("\r\n");
        #endif
    }
}
```

Video of Waldo Input

Circuit Diagram



Sensitivity

Range of Motion

The head of the dog is free to rotate in both the CCW and CW directions. Hence, it's resting state value is 150 degrees, and it can go to both the minimum (0 degrees) and maximum (300 degrees).

The paw of the dog is free to rotate only in the CW direction. Hence, it's resting value is 0 degrees, and it can go to the maximum (300 degrees), given there are no obstructions.

Noise Sources

Potential sources of noise in the ADC values would be loose wiring connections between the potentiometer and the Teensy, as well as gravity sliding the mechanical components, causing the potentiometer to change resistances.

3.2 Waldo Output - TBD