Desi Tomaselli, David Koval

ECE-218: Embedded Microcontrollers

Project 2 – Control of Servo Motors

February 4, 2018

**Introduction:**

**Objectives:**

The goals of project 2 in our Embedded Microcontrollers class was to create an open-loop control system with a potentiometer that controls two servo motors, one positional, and one continuous motor. A slider switch was provided for us to use to switch between positional control and speed/direction control. The two motors are supposed to operate in their full range with response to the potentiometer. When the positional motor is not selected, it holds its last position and the continuous servo should stop. There were several suggested milestones for my partner and I to achieve before attempting the main objective. The first milestone was to use the potentiometer to control one servo motor. The second milestone was to add the slider switch. In one switch position the servo is controlled with the potentiometer and in the other position, the potentiometer does not affect the servo – the servo remains at the previous position (or speed/direction) it was at when the switch was moved. The third and last milestone was to add the second servo and complete the project objective.

**Hardware:**

The hardware for this project consisted of a PIC24 Microchip, a breadboard, two servo motors (one positional and one continuous), a potentiometer, a slider switch, some resistors, wires and a computer to connect all the hardware to. The slider switch was connected to pin 26 (RB15) on the Microchip. The potentiometer was connected to pin 3 (AN1). The continuous servo was connected to pin 4 (RB0) and the positional servo was connected to pin 11 (RB4). The computer was connected to the PIC24 via USB and it was used to install the code onto it. All the major components of the circuit can be seen in Figure 1.

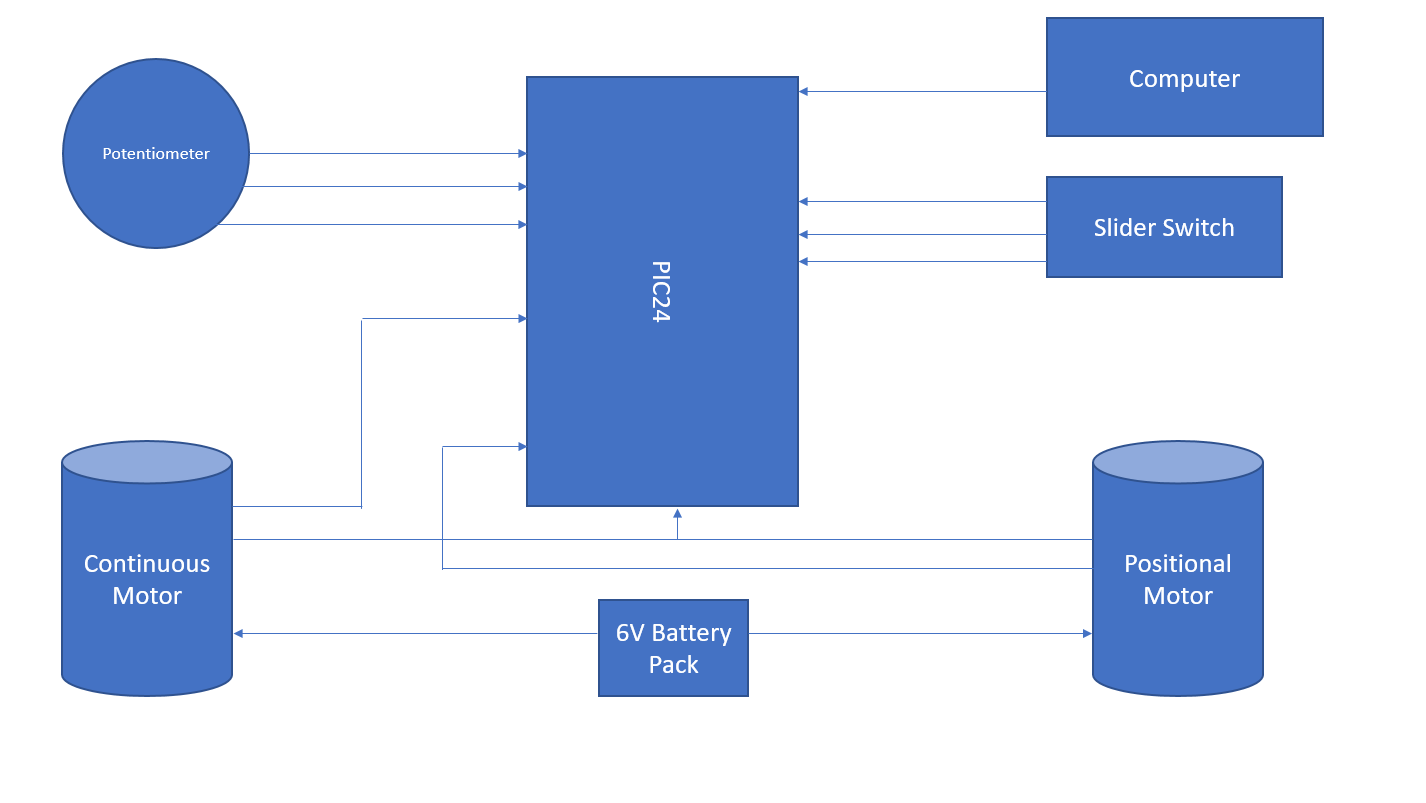


Figure 1: Block Diagram of Circuit

The potentiometer and slider switch were used as inputs whereas the two servos (the continuous and positional) were outputs. However, the potentiometer utilized an analog signal, whereas the slider switch and the PWM (Pulse-width modulation) for the servos were both digital signals. In our circuit, the potentiometer was used to change either the position or the speed and direction of the servo motors. The slider switch was used to switch between the two servos and what the potentiometer controlled. The 6-volt battery was used to power the two servo motors and the PIC24 powered the potentiometer and slider switch with 3.3-volts. With all those components hooked up and wired, we connected the PIC24 to the computer via USB and downloaded our code onto the microcontroller.

**Software:**

Declared constants for the pulse width modulation for continuous and positional servos:

C\_MIN 203 – 1.3ms pulse width for continuous servo

C\_MAX 266 – 1.7ms pulse width for continuous servo

C\_MID 234 – 1.0ms pulse width for continuous servo

C\_pulse\_width – Pulse width of the continuous servo

P\_MIN 117 – 0.75ms pulse width for positional servo

P\_MAX 351 – 2.25ms pulse width for positional servo

P\_MID 156 – 1.5ms pulse width for positional servo

P\_pulse\_width – Pulse width of the positional servo

Declared constants for the slider switch:

SW PORTBbits.RB15 – Declares the slider switch at pin 26 (RB15)

SW\_CONTINUOUS() (SW == 1) – Sets the slider at position 1 to control the continuous servo motor.

SW\_POSITIONAL() (SW != 1) – Sets the slider at the other position to control the positional servo motor.

Declared constant for the potentiometer value:

adcvalue – Potentiometer value

Global Functions:

Scale – Converts the adcvalue to a scalable value for the pulse width with relation to the max and min values of the continuous and positional servo motors.

Setup:

Initialize ports RB0, RB4, RB15 and AN1.

Enable timer 2 interrupts.

Configures the clock to 40MHz, heartbeat, timer 2, and output compare modules 1 and 2 for continuous servo and positional servo.

Main Loop:

Reads the analog value at pin AN1.

If SW\_POSITIONAL is selected:

Set C\_pulse\_width = C\_MID

Set P\_pulse\_width = adcvalue after scaling it appropriately with the positional servo’s range

Delay by 50ms

If SW\_CONTINUOUS is selected:

Set C\_pulse\_width = adcvalue after scaling it appropriately with the continuous servo’s range

Delay by 50ms

**Results:**

The circuit and servo motors worked as expected. We were able to control the servo motors by using the slider switch to select between the two and the potentiometer to change the speed and direction of them. However, unfortunately, the potentiometer did not completely work how we expected for it to. The analog readings from the potentiometer were 4096 when it was turned half way, which means, the analog value should be 2047.5, and not 4096. Other than that, the entire circuit worked. We managed to get the two servos turning. The full circuit can be seen in Figure 2.

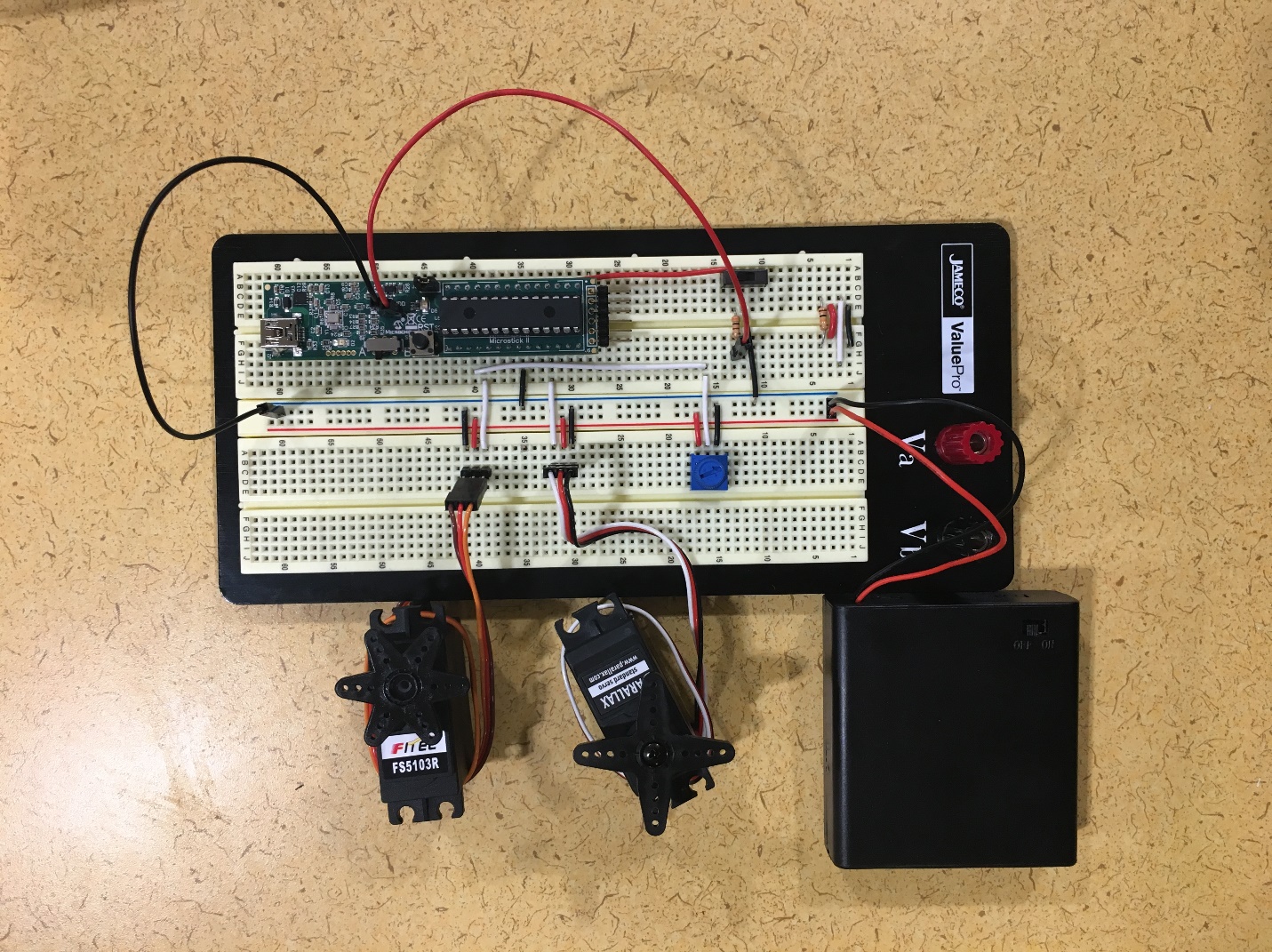


Figure 2: Full Circuit with Servo Motors

**Conclusion:**

The overall results satisfied the expected goals of this project. The entire circuit worked as expected based on our wiring and coding. With persistence and confidence, we managed to make our circuit work within a few hours. The slider switch was switching between the control of the two servos motors using the potentiometer and the potentiometer was moving the positional servo freely. Some problems that we encountered were that the potentiometer was not accurate. We found the readings of the analog value to be 4095 when the potentiometer was only at half of its resistance. The reading should have been 2047.5 at half of its resistance, which means that it limited our range of motion. The only way to really improve our project would be to have a better, more accurate potentiometer since we were not able to get the full range of motion out of our servos.

**Appendix:**

Source Code:

/\*

\* David Koval, Desmond Tomaselli

\* Professor Traver

\* Project 2

\* Feb. 5, 2017

\*/

/\*\*\*\*\*\*\*\*\*\*\* COMPILER DIRECTIVES \*\*\*\*\*\*\*\*\*/

// #include for textbook library header files

#include "adc.h" // header for simple adc library

#include "pic24\_all.h"

// #defines for handy constants

#define PWM\_PERIOD 3125 //20ms PWM period with clock cycle of 6.4us from 1:256 prescale

#define C\_MIN 203 //1.3ms (min) pulse width for rotation servo

#define C\_MAX 266 //1.7ms (max) pulse width for rotation servo

#define C\_MID 234 //1.0ms pulse for rotation servo

#define P\_MIN 117 //0.75ms (min) pulse width for positional servo

#define P\_MAX 351 //2.25ms (max) pulse width for positional servo

#define P\_MID 156 //1.5ms pulse for positional servo

#define SW PORTBbits.RB15 //Set switch to pin 26 (port RB15)

#define SW\_CONTINUOUS() (SW == 1) //Switch is reading a high value

#define SW\_POSITIONAL() (SW != 1) //Switch is reading a low value

/\*\*\*\*\*\*\*\*\*\*\* GLOBAL VARIABLE AND FUNCTION DEFINITIONS \*\*\*\*\*\*\*/

uint16\_t C\_pulse\_width; //pulse width for Continuous motor

uint16\_t P\_pulse\_width; //pulse width for Positional motor

uint16\_t adcvalue; //potentiometer value

// Configures Output compare module 1 for continuous Ratation Servo

void configOC1() {

T2CONbits.TON = 0; //Turns timer 2 off.

CONFIG\_OC1\_TO\_RP(RB0\_RP); //Maps the OC1 output to the remappable pin, RB0.

OC1RS = 0; //Clears the RS register

OC1R = 0; //Clears the R register

OC1CONbits.OCTSEL = 0; //Sets the output compare module to use Timer 2 as the clock source.

OC1CONbits.OCM = 0b110; //Sets it to operate in PWM mode with fault pin disabled.

}

// Configures Output compare module 2 for positional Servo

void config0C2(void){

T2CONbits.TON = 0; //Turns timer 2 off.

CONFIG\_OC2\_TO\_RP(RB4\_RP); //Maps the OC2 output to the remappable pin, RB4.

OC2RS = 0; //Clears the OC2RS register

OC2R = 0; //Clears the OC2R register

OC2CONbits.OCTSEL = 0; //Sets the output compare module to use Timer 2 as the clock source.

OC2CONbits.OCM = 0b110; //Sets it to operate in PWM mode with fault pin disabled.

}

//Configures Timer 2

void configTimer2() {

T2CON = 0x0030; //Configs timer 2, with presacle 1:256

PR2 = PWM\_PERIOD; //Sets period

TMR2 = 0; //Clears the timer 2 register.

\_T2IF = 0; //Clears the flag, T2IF

}

//Scales adcvalue to proportional value in the range of p\_min and p\_max

uint16\_t scale(uint16\_t u16\_x, uint16\_t p\_min, uint16\_t p\_max){

return (uint16\_t)(((u16\_x/4095.0)\*(p\_max-p\_min)) + p\_min);

}

//Configures the timer 2 interupt and loads its registers

void \_ISR \_T2Interrupt(void) {

OC1RS = C\_pulse\_width; //Load OC1RS register with Continuous pulse width

OC2RS = P\_pulse\_width; //Load OC2RS register with Positional pulse width

\_T2IF = 0; //Clears the Timer 2 interrupt flag, the last instruction in ISR

}

/\*\*\*\*\*\*\*\*\*\* MAIN PROGRAM LOOP\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main ( void ) //main function that....

{

/\* Define local variables \*/

/\* Call configuration routines \*/

configClock(); //Sets the clock to 40MHz using FRC and PLL

configHeartbeat(); //Blinks the LED on RA1

configOC1(); //Configures output compare module 1 for Continuous motor

config0C2(); //Configures output compare module 2 for Positional motor

configTimer2(); //Configures timer 2

/\* Initialize ports and other one-time code \*/

TRISBbits.TRISB15 = 1; //Map pin 26 (RB15) to switch

\_TRISB15 = 1; //Set pin 26 (RB15) to be an input pin

\_CN11PUE = 1; //Set pull up resistor for pin 26 (RB15)

initADC(AN1); //Initialze pin AN1 for analog reading of Potentiometer

\_T2IE = 1; //Enables the timer 2 interrupts

T2CONbits.TON = 1; //Turns timer 2 on

/\* Main program loop \*/

while (1) {

adcvalue = readADC(AN1); //reads in value from port AN1 (voltage from resistance of Potentiometer)

if(SW\_POSITIONAL()){ //If switch is set to ground then control Positional motor

C\_pulse\_width = 0; //Set continuous servo motor PW to 0

P\_pulse\_width = scale(adcvalue, P\_MIN, P\_MAX); //Scale potentiometer voltage to a PW that corresponds to a position between 0 and 180 degrees

DELAY\_MS(50); //50 ms delay

}

if(SW\_CONTINUOUS()){ //If switch is set to high then control Continuous

C\_pulse\_width = scale(adcvalue, C\_MIN, C\_MAX); //Scale potentiometer voltage to PW that changes speed of continous motor

DELAY\_MS(50); //50 ms delay

}

}

}

See Figure 3 for full schematic of circuit.

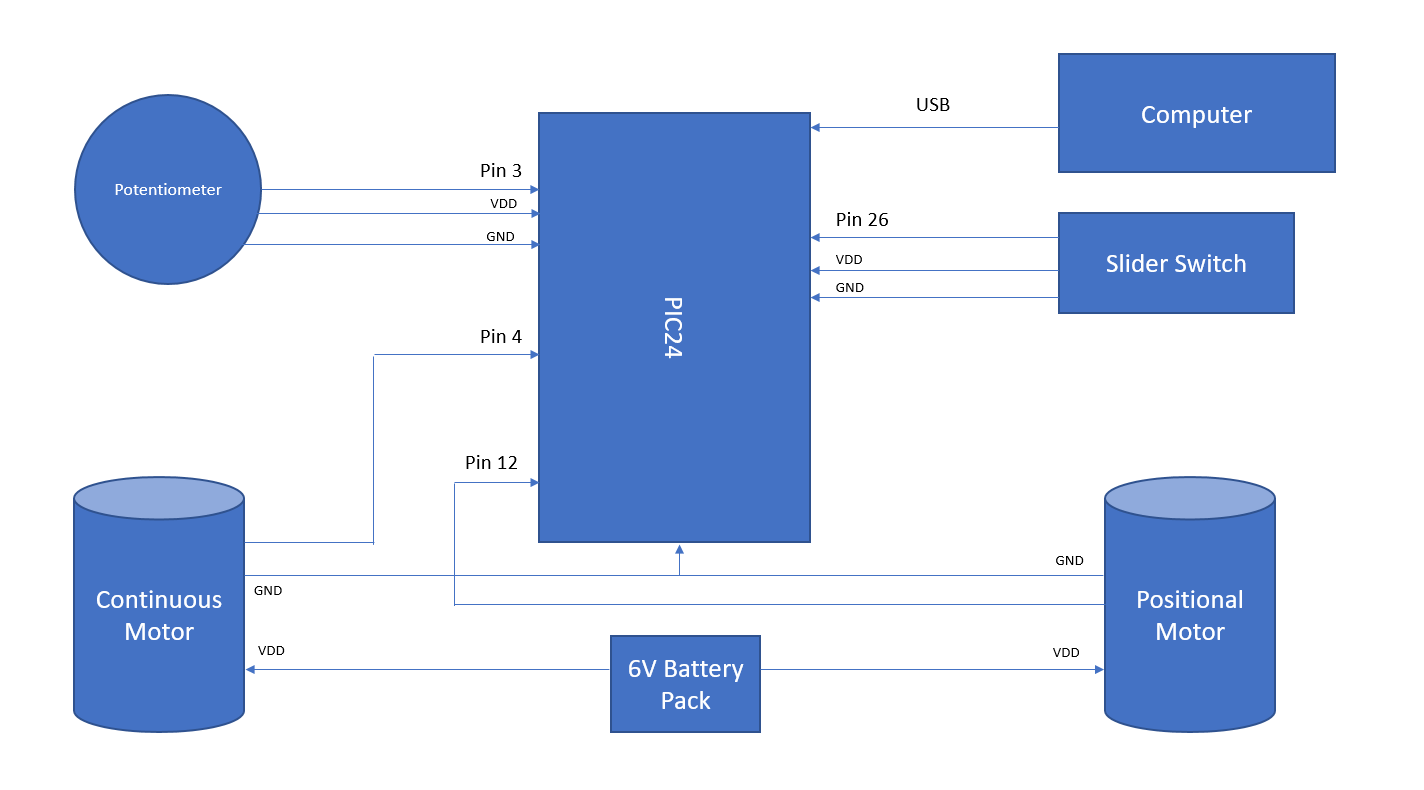


Figure 3: Full Schematic of Circuit