Robotic Telescope Game

Chirag Sirigere & Austin Choi

ECE230 Introduction to Embedded Systems

Winter 2020-2021

Instructor: Dr. Michael Jo

Revised: 02/22/2021

Table of Contents

Introduction & Overview of the Robotic Telescope Game

Introduction

During our project brainstorming session, we knew that robotics is a growing industry and thought of ways to implement our knowledge of embedded systems in some robotics applications. We immediately had an idea that we wanted to pursue: robotic telescopes.

Project Overview

In this telescope implementation game, we will have two modes. The first mode uses inputs from both a potentiometer and push buttons to rotate two servo motors. These inputs help the user maneuver the telescope to find constellations around the room. Each servo motor will be connected to one input for each axis of rotation. The user can rotate the potentiometer to rotate the servo motor left or right and press a button to rotate another servo motor up or down by ten-degree steps. When a servo motor reaches the limit for the axis of rotation, an LED will light red and the player will no longer be able to move in that direction. There will be an LCD that will display the degrees that the player has turned for each servo motor. The objective of the game is to locate all constellations before the game resets after 30 seconds.

The other mode will take inputs from a keypad and use those inputs to go to a predetermined constellation position. For example, if the player presses ‘1’ on the keypad, each servo motor will rotate and point the telescope in the direction of constellation ‘1’. If the player presses ‘2’, the telescope will turn to constellation 2, and so forth.

Meeting the Advanced Project Requirements

The requirements for the project completion includes all of the following:

1. The project must use the MSP432P401R (or other microcontroller with approval)
    a. Additional microcontrollers may be used in project if needed (parallel process)
2. The embedded circuit must include at least one sensor (input)
    a. Pushbutton, analog signal, serial communication input, etc.
3. The embedded circuit must include at least one actuator (output)
    a. LED, LCD, motor, etc.
4. The embedded project must make use of at least one timer
5. The embedded project must make use of at least one interrupt

The requirements for the advanced project includes all of the above and the following:

1. Includes at least 6 unique sensors/actuators
2. Uses at least two peripherals that were not implemented in a lab exercise OR uses them in a significantly different configuration (e.g. SPI configuration rather than I2 C)

The final project that the team has created, does meet the Advanced Projects Requirements as described below:

1. The project uses the MSP432P401R microcontroller
2. The project uses at least six sensors or actuators
    a. LEDs
    b. Servo motors
    c. Keypad
    d. LCD
    e. Potentiometer
    f. Push buttons
3. The project uses at least one timer
    a. Watchdog timer
4. The project uses at least one interrupt
    a. Timer_A interrupt
5. The project uses at least two peripherals
    a. Watchdog timer
    b. Keypad
    c. Timer_A

Operation of the Telescope

Objective

There will be six constellations to locate in the entire thirty seconds provided to the user. The user uses the potentiometer and the two buttons to control the movement of the servo motors up, down, left, or right. The goal of the user is to rotate the servo motors so that the telescope points in the direction of a constellation. The angles displayed on the LCD helps the user to fix the servo motor angles. A constellation is considered located when the user is within one degree of the rotations displayed on the LCD. The game is over when the user fully locates all constellations before the game resets or the game resets before the user fully locates all constellations.

Rotating the Telescope Left/Right

To rotate the telescope left and right, the user must use the potentiometer which controls the servo motor on the bottom.

Tilting the Telescope Up/Down

To tilt the telescope up and down, the user must use the two buttons which control the servo motor on the side. The left button tilts the telescope down and the right button tilts the telescope up.

Function of LED While Operating

When the user rotates one or both of the servo motors to the servo motor maximum angles, the red LED will brighten indicating that the maximum angles have been reached.

Function of LCD While Operating

The LCD displays two pieces of information: the angle of the tilt and the angle of the rotation. The rotation and tilt angles can be correlated to the x- and y-axis rotations of the telescope. Using this information, the user can adjust the telescope to point at the correct constellation.

Function of Keypad Mode

This mode is implemented as if the robot or operator has memorized the locations of the constellations and all the user needs to do is press a number on the keypad for the telescope to adjust itself to point to the correct constellation.

Hardware Design and Implementation

Power Supply

The power supply for the robotic telescope is a laptop so that the user can simply press the debug button in the compiler window and start the game.

Circuit Diagram

Figure 1: The LCD wiring diagram is shown below

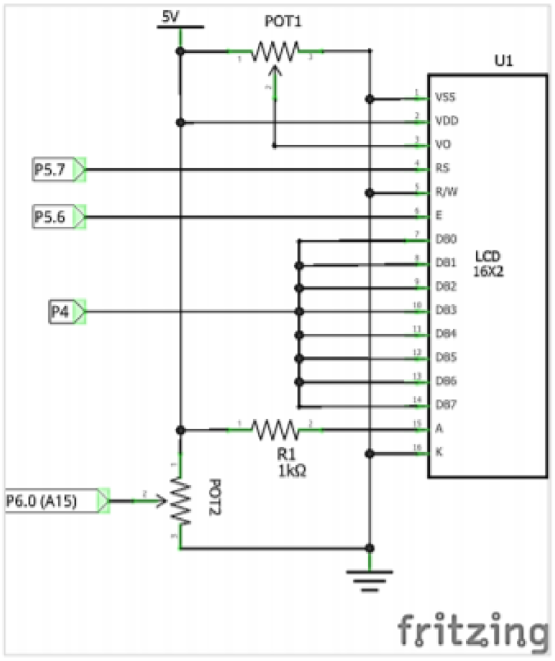
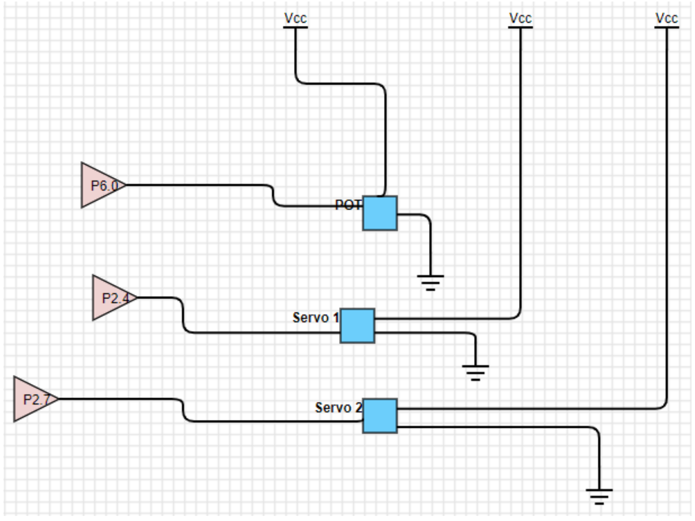


Figure 2: The wiring diagram for the servo motors along with the potentiometer is shown below



Pin Assignments

Originally, the idea was to use individual buttons to control the servo motor to rotate the telescope left and right, but it would be easier to implement the left and right movements with the on-board buttons on the microcontroller.

The following pin assignments are taken from the circuit diagram above:

P6.0	Potentiometer
P2.4	Servo motor Left/Right
P2.7	Servo motor Up/Down
P4	LCD: DB0-7
P5.7	LCD: RS
P5.6	LCD: E

Telescope Box

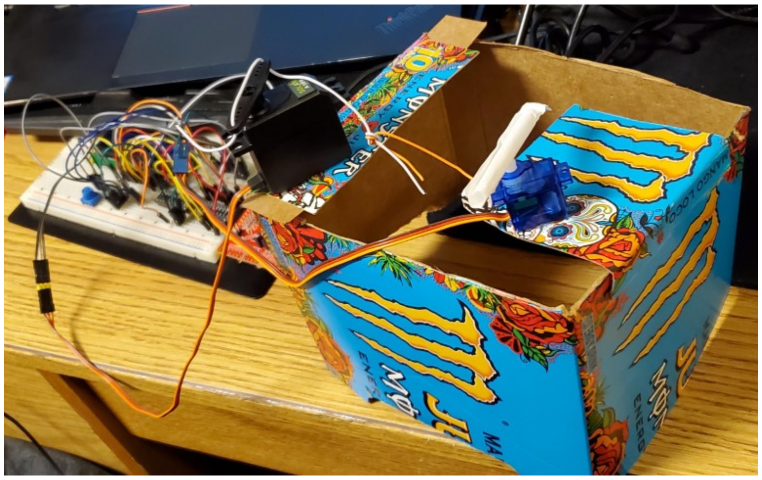


Figure 3: The entire setup of the project with the box

The telescope box is simply made from a cardboard box and the servo motors are connected to a straw. The telescope is the straw with the lens fitted on each end of it. The straw is connected directly to one servo to control one axis of rotation. The other servo is connected to the platform of the original servo. When it is rotated one direction, a string gives slack to the platform, rotating the platform down. When the servo is rotated the other direction, a string gets tighter, raising the platform up.

The telescope box itself needs to be stable enough so that the platform only rotates and doesn’t fall off. The telescope box has to hold the weight of both the servos and not completely collapse.

However, we did want the telescope box to be light enough so that the servo would not have to be strained in order to rotate the box.

We ended up using the carcass of an energy drink box. We figured that it was lighter than conventional cardboard, and still stiff enough to maintain form. This worked perfectly for us. The only small issue was that the box was slightly too stiff. We fixed this by cutting up the box and re-taping it together. This way, the axis of rotation is more loose. Originally, the weight of the servo itself was not enough to drop the box, but now the axis of rotation of the box is loose enough as to where it can freely rotate.

Hardware Images

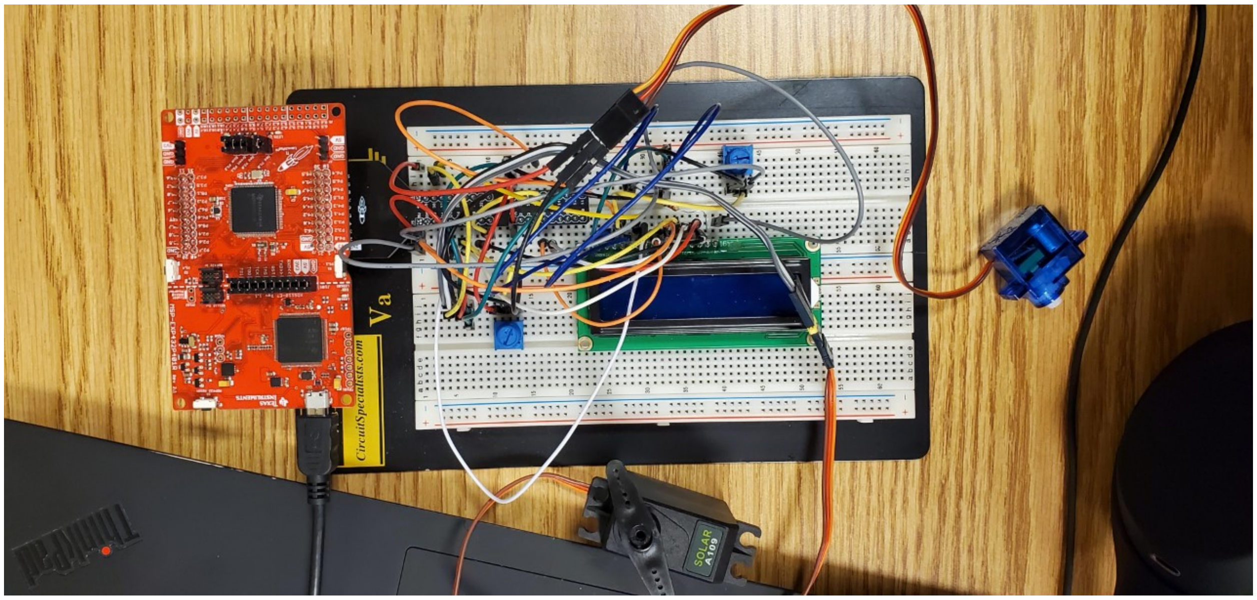


Figure 4: Top view of the project showing everything

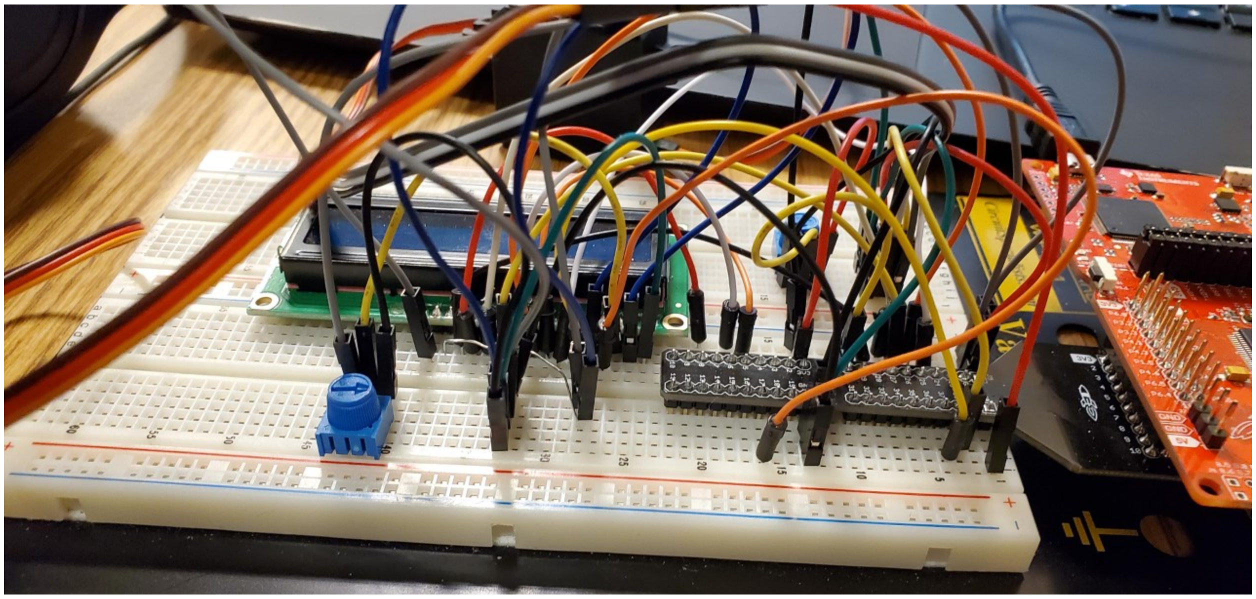


Figure 5: Front view of the project showing the LCD potentiometer, microcontroller, servo motor potentiometer and LCD behind wiring

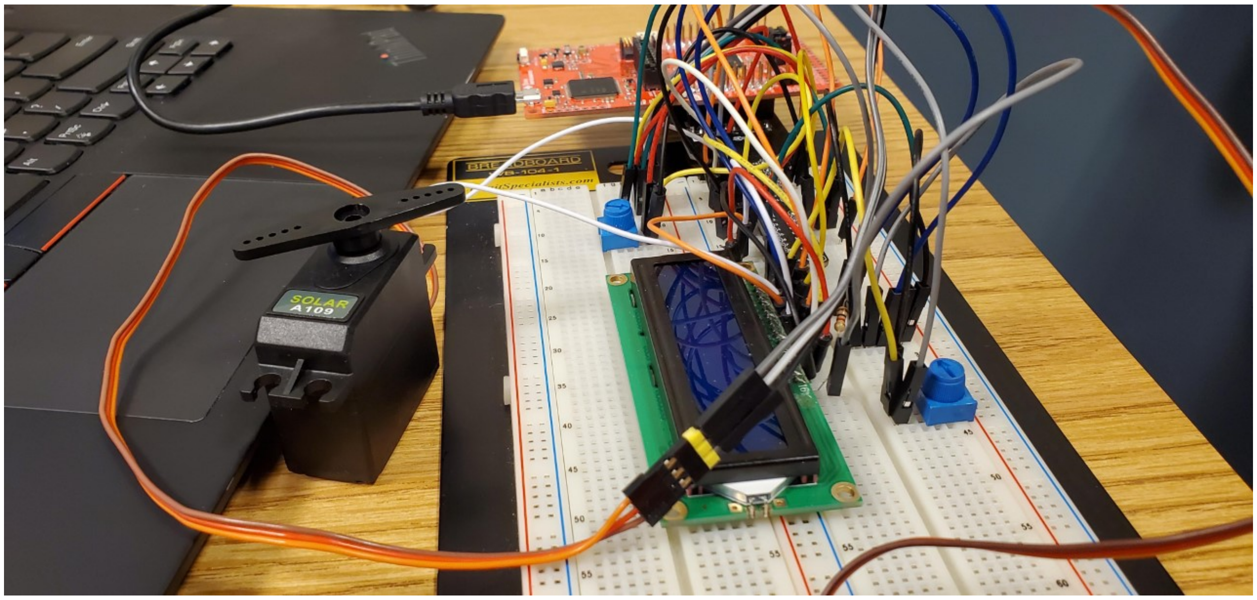


Figure 6: Side view of the project showing the LCD, LCD potentiometer, servo motor potentiometer, and microcontroller behind wiring

Software Design and Implementation

Flow Chart

Shown below is the flowchart that describes the main game with the user interaction between the potentiometer, push buttons, and servo motors.

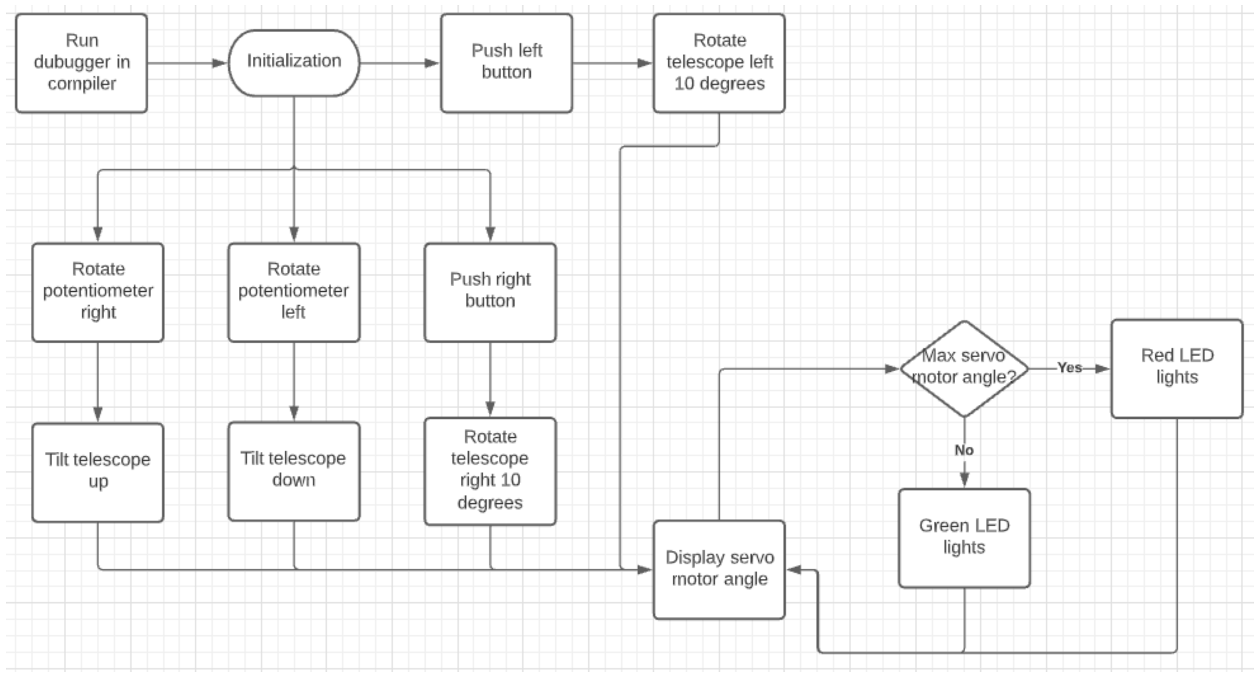


Figure 6: Software flowchart of the user driven game mode

Servo Motor Programming

The way the servo motor is programmed to rotate is simple. The voltage outputted by the potentiometer to the pin on the microcontroller is converted to a digital value and the digital value is again converted to a PWM signal that the servo motors can read. This can be seen in the code snippet below.

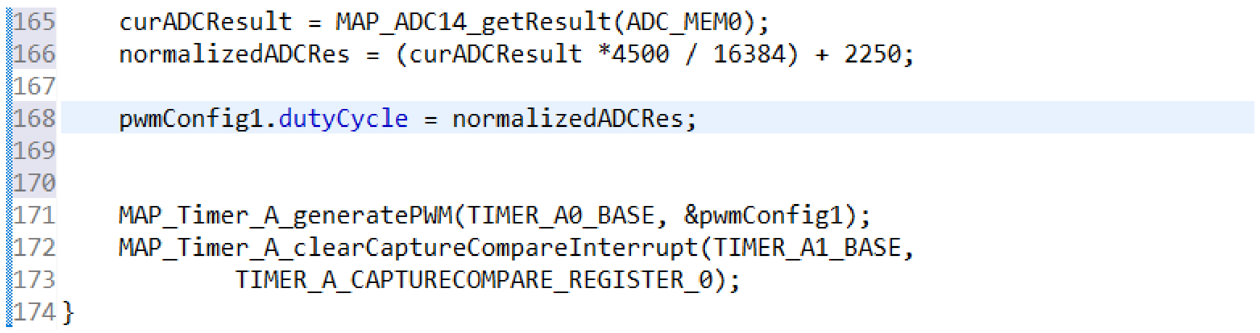
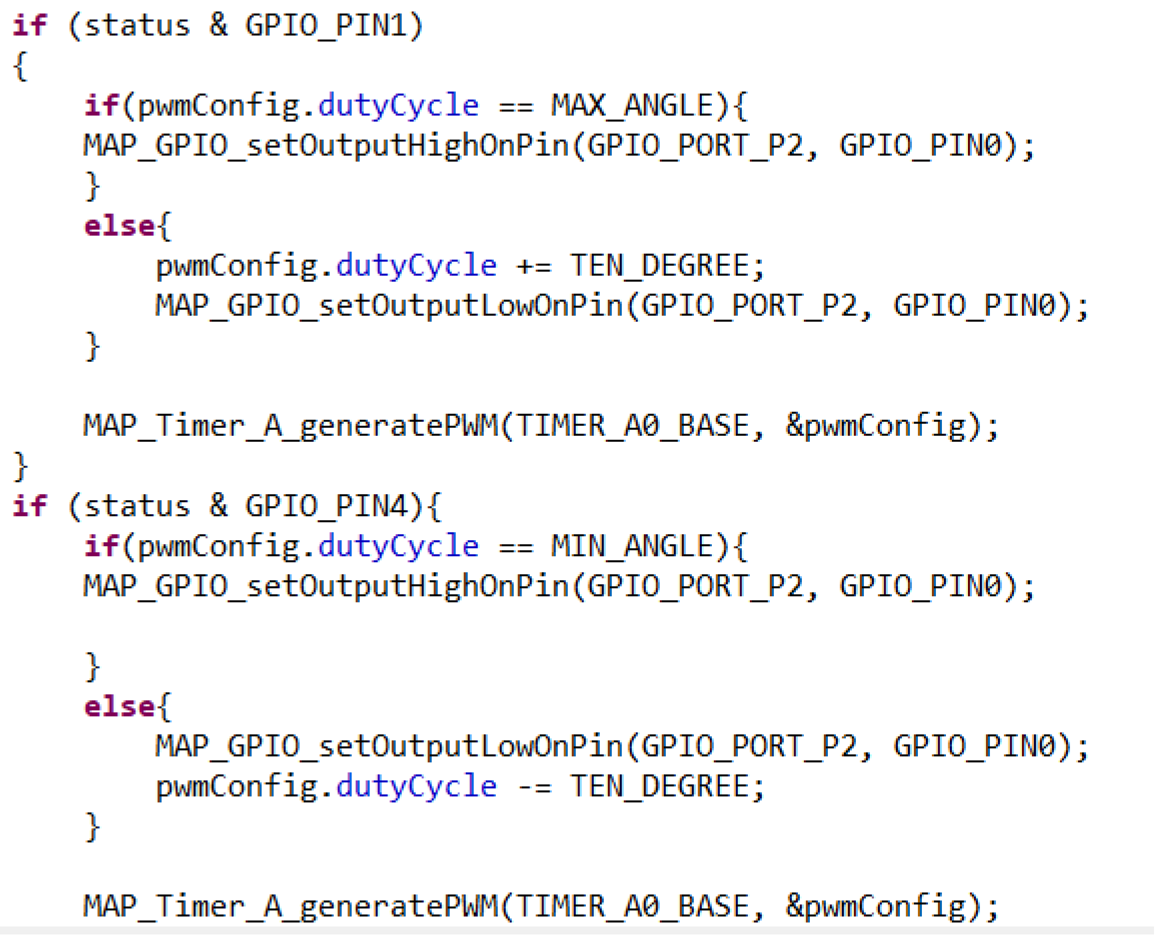

```
    curADCResult = MAP_ADC14_getResult(ADC_MEM0);
    normalizedADCRes = (curADCResult *4500 / 16384) + 2250;

    pwmConfig1.dutyCycle = normalizedADCRes;


    MAP_Timer_A_generatePWM(TIMER_A0_BASE, &pwmConfig1);
    MAP_Timer_A_clearCaptureCompareInterrupt(TIMER_A1_BASE,
            TIMER_A_CAPTURECOMPARE_REGISTER_0);
}
```

Figure 7: Image of the code snippet containing the ADC to PWM signal generation

We found that the best way to do this was to use a timer interrupt to check the value of the potentiometer once a second. For some reason, when we tried to have it within the while loop, the servo would only update once and then it would never update again until another interrupt was introduced. By using a timer interrupt, we avoided this problem as the servo is forced to update every second.

The servo that rotates the vertical axis is controlled by a button press. When a specified button is pressed, there is a button interrupt that changes the rotation of the servo. If the limit of the servo is hit in any rotation, a red light will turn on that shows that the rotation has been hit.

A snippet of this code is below.



```
if (status & GPIO_PIN1)
{
    if(pwmConfig.dutyCycle == MAX_ANGLE){
    MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P2, GPIO_PIN0);
    }
    else{
        pwmConfig.dutyCycle += TEN_DEGREE;
        MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN0);
    }

    MAP_Timer_A_generatePWM(TIMER_A0_BASE, &pwmConfig);
}
if (status & GPIO_PIN4){
    if(pwmConfig.dutyCycle == MIN_ANGLE){
    MAP_GPIO_setOutputHighOnPin(GPIO_PORT_P2, GPIO_PIN0);

    }
    else{
        MAP_GPIO_setOutputLowOnPin(GPIO_PORT_P2, GPIO_PIN0);
        pwmConfig.dutyCycle -= TEN_DEGREE;
    }

    MAP_Timer_A_generatePWM(TIMER_A0_BASE, &pwmConfig);
```

Figure 8: Image of the code snippet containing the rotation of the servo motor based on the push button input

Keypad Interrupt

For the keypad, we wanted to make it so that the keypad will disable the potentiometer and push button inputs. This way, the servos do not get confused as they do not have two things that are giving it an input. The keypad buttons work in the same way that the other push buttons do.

When a keypad button is pushed, the system will hit an interrupt. This interrupt will disable inputs from other sources. They push buttons will then give each of the servos a designated pwm value.

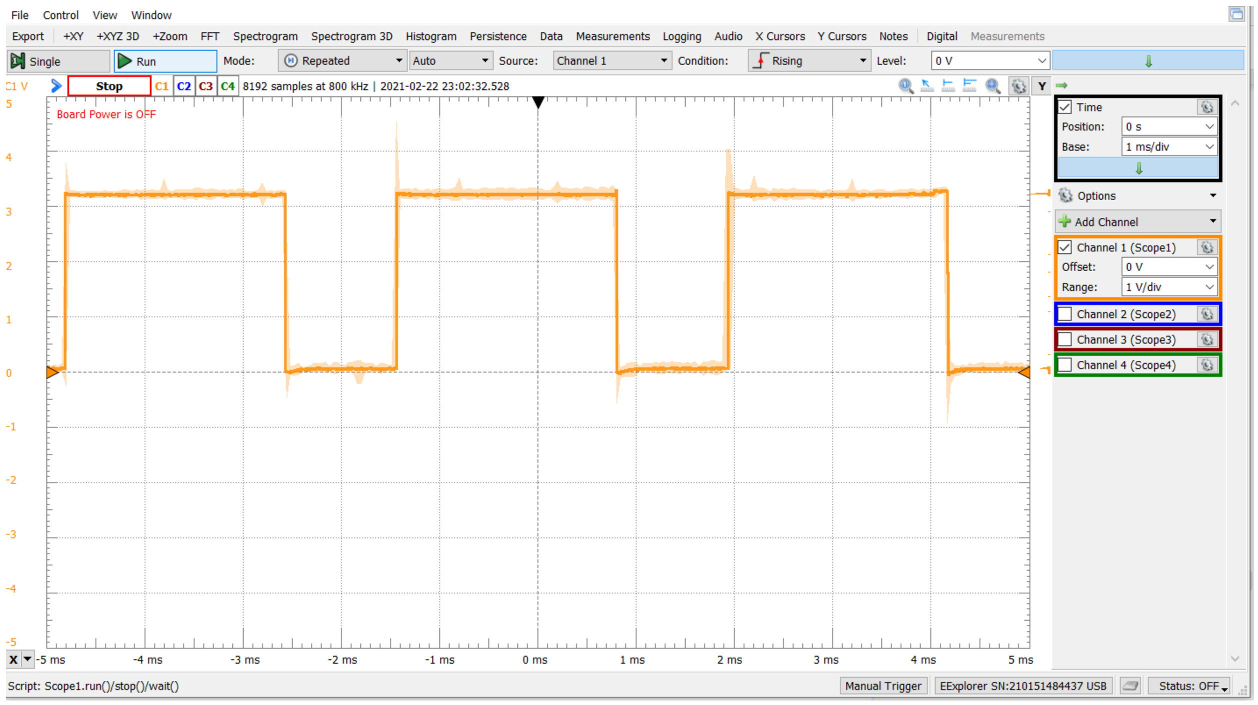
Testing Procedure of the Telescope

Individual components were tested to check if they worked and were connected to either buttons or potentiometers

Servo Motors

For the servo motors, we followed steps that were similar to the steps we followed to verify our project plans. We tested the servo motors with an oscilloscope and showed their pwms. We showed each of the extremes for the servos. Because the inputs of the servos are essentially the same, it was only necessary to test one servo. We chose to test the vertical-axis rotation servo as it seemed to be slightly more stable.

Figure 9: The PWM below is of the max duty cycle of the servo.



Mode: Repeated | Auto | Source: Channel 1 | Condition: Rising | Level: 0 V

8192 samples at 800 kHz | 2021-02-22 23:02:32.528

Time — Position: 0 s, Base: 1 ms/div

Channel 1 (Scope1) — Offset: 0 V, Range: 1 V/div

Channel 2 (Scope2)

Channel 3 (Scope3)

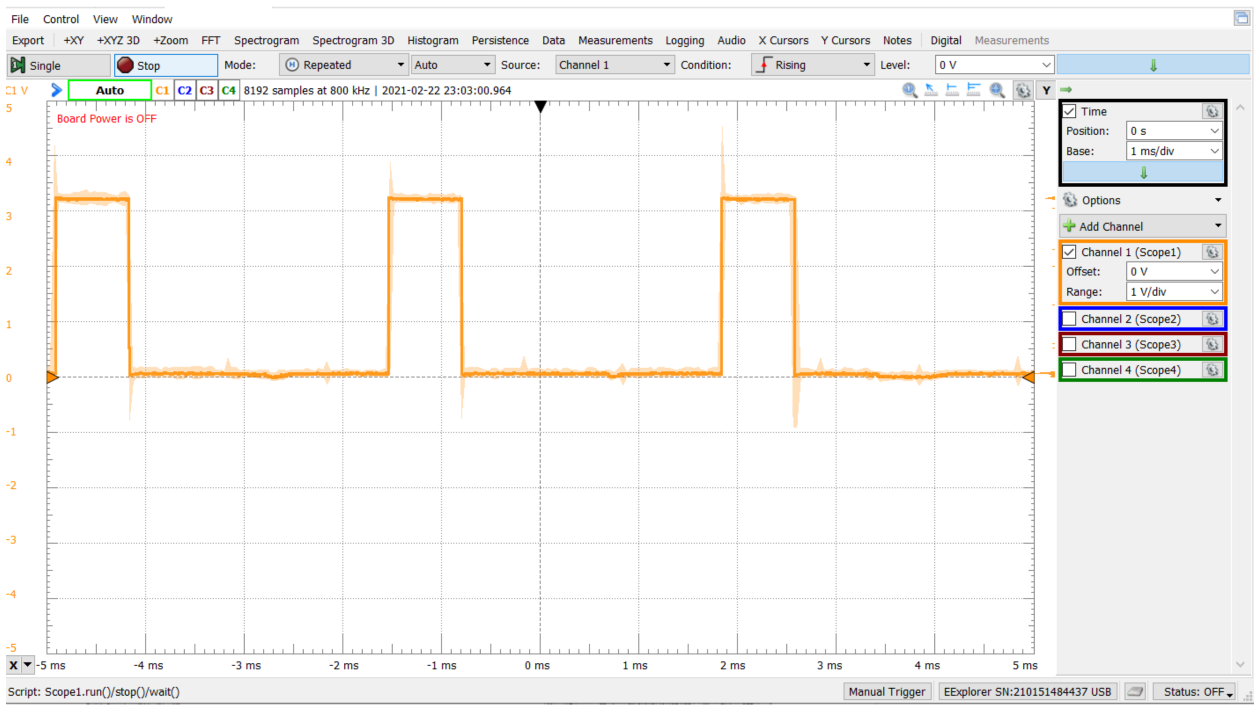
Channel 4 (Scope4)

Script: Scope1.run()/stop()/wait()

EExplorer SN:210151484437 USB

Status: OFF

Figure 10: The PWM below is of the minimum duty cycle of the servo


File Control View Window
Export +XY +XYZ 3D +Zoom FFT Spectrogram Spectrogram 3D Histogram Persistence Data Measurements Logging Audio X Cursors Y Cursors Notes Digital Measurements
Single
Stop
C1 V
Auto
C1 C2 C3 C4
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5 ms
-4 ms
-3 ms
-2 ms
-1 ms
0 ms
1 ms
2 ms
3 ms
4 ms
5 ms
Options
Add Channel
Manual Trigger

Mode: Repeated Auto Source: Channel 1 Condition: Rising Level: 0 V

8192 samples at 800 kHz | 2021-02-22 23:03:00.964

Board Power is OFF

Time Position: 0 s Base: 1 ms/div

Channel 1 (Scope1) Offset: 0 V Range: 1 V/div

Channel 2 (Scope2)

Channel 3 (Scope3)

Channel 4 (Scope4)

Script: Scope1.run()/stop()/wait()

EExplorer SN:210151484437 USB

Status: OFF

Potentiometer to Servo Motors

We tested the potentiometer by seeing the values that were shown in our debugger. We knew what our debugger values should be around, and we verified that these values were correct.

LCD

We tested to make sure that the debugger values were identical to the values displayed on our LCD. We also saw the LCD and verified that it worked as intended.

LED

We tested the LEDs by going to the limits of each of the rotations. Once we hit the rotation and the servo stopped rotating, we verified to see if the LEDs had turned on. When we went to each of the extremes, we saw and made sure that the LED is now red.

Keypad to Servo Motors

Bill of Materials

Component Name	Description	Unit Price/Acquired	Quantity
LED	Servo motor maximum angle	Acquired	1
LCD	Servo motor angle display	Acquired	1
Potentiometers	Control of servo motor	Acquired	2
Keypad	Automatic constellation locator	Acquired	1
Cardboard	Stand, tube, and container box	Acquired	1
Servo Motors	Telescope rotation	Acquired	2
Lens	Cardboard tube	Acquired	2
Breadboard	Overall circuitry	Acquired	1
Laser pointer	Constellation locator	Acquired	1

Future Plans & References

Future Plans

The future plans for this project are vast and can be applied to so many problems, but the first improvement would be to make the project adaptable to a more stable battery source. We had one servo using the 5V output and the LCD display which also used the 5V output. When we changed the PWMs, we observed that there were occasional flickers that happened. We think this is because the servo motors are taking too much power. We could fix this by using more 5V outputs or by using a more stable 5V output.

As it stands, our game serves more as a pointer than an actual telescope. In the future, we can use this as a beta version of a telescope. On a larger scale, we can attach lenses and a larger body to make an actual functioning telescope. We can also use a larger and more stable body. Our vertical axis of rotation was lackluster, so in the future we can improve upon that design.

If we go more of the “game” route, we can put the telescope inside of a large box and have the player point at celestial objects that are painted on the walls of the box. Once an object is pointed at, the LED can flash green or the LCD can display that the user has found the object. The watchdog timer can be implemented so that the user has a select amount of time before the game ends and the system shuts down.

All in all, our project can be used as a beta model for many different directions. We can use what we learned from this project and apply it to a variety of different projects in the future. Our project used a wide breadth of knowledge taken from many different times of this course and put them all together.

References

We would like to thank Dr. Jo for providing us with an amazing instruction video and exercise to learn about LCDs. The circuit diagram in the Hardware Design and Implementation section of this report (Figure 1) is taken from the in-class lesson on LCD (Lesson 6-4) and the exercise (Exercise 6-2) as well as Project 5.