JOYSTICK CONTROLLED SERVO MOTOR USING A PIC MICROCONTROLLER

Author: Taahir Kolia (2423748)

School of Electrical & Information Engineering, University of the Witwatersrand

Abstract: The aim of the project is to create a synchronous system that accurately allows a joystick to control the movement of a servo motor. The ADC and PWM modules of the microcontroller are configured to control the components. The values stored in the ADC registers play a vital role in varying the pulse width of the PWM which allows the joystick to control the servo. The outcome of the project is that joystick can control the servo but not as accurately as expected.

1. INTRODUCTION

The aim of this project is to use a Curiosity Nano microcontroller (PIC16F18446) to develop and implement an integrated system that comprises of a servo motor and an analog joystick. The position of the analog joystick is required to be synchronized with the movement of the servo of the joystick. The servo motor is required to be centered when the joystick is in the default position. When the joystick is moved to the left or right, the servo should move in the respective directions.

The report explains how the different components are used and how the different modules of the microcontroller are configured for the servo motor and joystick to be operable.

2.BACKGROUND RESEARCH

2.1 Servo motors:

The servo motor has three pin connections, a ground pin (brown), a 5V power supply pin (red) and a control signal pin (orange).

The servo motor has a 0-180° range of movement and can either turn 90° clockwise or anticlockwise [1]. The movement is determined by a pulse width modulation (PWM) that is sent to the control signal pin of the servo motor [1]. The period of the PWM required for the servo is 20ms [1]. A pulse width of 1ms rotates the servo anti clockwise to 0°, whereas a pulse width of 1.5ms and 2ms corresponds to a rotation to 90° and 180° respectively [1]

2.2 Analog joystick:

The joystick consists of two 10k potentiometers which interpret the joystick's position on the 2-dimenionsal axis and convert it into analog readings that can be processed by the microcontroller. The changes in the resistance of the potentiometers when the joystick is moved can be processed using the analog-to-digital converter (ADC) of the microcontroller to determine the position of the joystick. [2]

The analog joystick has five pin connections: a ground pin, a 5V power supply pin, a VRx pin, a VRy pin, and a SW pin. The VRx and VRy pins output the values of the joystick in the horizontal and vertical directions whilst the SW outputs a high or low signal from when the joystick is pushed [2].

3. PLANNING

3.1 Flow Diagram

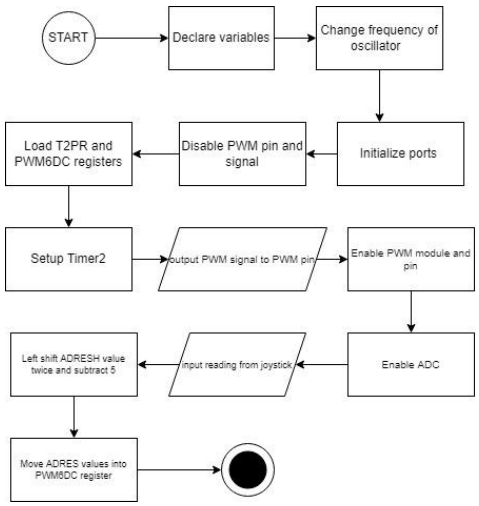

Figure 1: Flow diagram of the code to configure the PWM and ADC modules of the microcontroller.

3.2Circuit Design

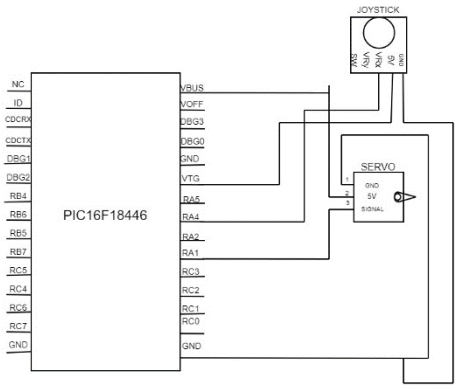

Figure 2: Circuit Diagram of the servo-joystick system

The signal pin of the servo motor is connected to a PWM pin (RA1), the 5V pin is connected to VBUS which supplies 5V and the ground pin is connected to GND.

The VRx pin of the joystick is connected to an analog pin (RA4), the 5V pin is connected to VTG which supplies 3.3V and the ground is connected to GND. Although the voltage supplied to the joystick is less than the recommended voltage, the joystick can still operate.

4. METHODOLOGY

4.1 Choosing the clock frequency

The frequency of the internal oscillator is set to 4MHz with a clock divider of two resulting in clock frequency of 2MHz. This is done using the configuration bits and the Oscillator Control Register1 (OSCCON1) [3].

4.2 Configuring the ADC

The ADC is configured by setting the ADC Enable bit, the ADC Continuous Operation bit and the ADC Conversion Status bit of the ADC Control Register 0 (ADCON0) [3]. The ADC Alignment Selection bit of the ADONC0 register is cleared to left-justify the ADC Result Register (ADRES) and the ADC Clock Selection bit is cleared to use the internal oscillator frequency clock [3]. The ADC clock frequency is then set to a quarter of the internal oscillator frequency clock using the ADC Clock Selection Register (ADCLK) [3].

4.3 Reading values from the ADC

To determine the value of the ADC when the joystick is centered or moved in either direction, eight LEDs are connected to the RC0-RC7 ports. The ADC values stored in the ADRESH register are moved into the LATC ports. Once the ports are enabled, the binary values of the ADC are displayed on the LEDs.

Table 1: ADC values when the joystick is in different positions

Position of joystick	ADRESH value (in decimal)
Extreme left	0
Centered	118
Extreme right	255

4.4 Configuring the PWM and Timer2 module:

The PWM pin and the PWM Configuration register (PWM6CON) register are disabled [3]. Thereafter. The Timer2 Period register is loaded with the calculated binary value needed to obtain a 20ms period [3]. Similarly, the PWM Duty Cycle (PWM6DC) register is loaded with the binary values needed to rotate the servo motor [3]. The PWM signal is then moved to the specified PWM pin. Lastly, the PWM module and pin are enabled [3].

The timer pre-scaler of 1:128 is selected and the Timer2 is enabled in the Timer2 Configuration register (T2CON). The timer clock source is set to a quarter of the frequency of the internal oscillator clock frequency [3].

4.5 Calculating the T2PR value

To obtain the period of the PWM, the following formula can be used [3]:

$$Period = (T2PR + 1) \cdot 4 \cdot Tosc \cdot (TMR2PrescaleValue) \quad ..(1)$$

Since the required PWM period, Tosc and pre-scaler values are all known, the value of T2PR is found to be equal to 77.

4.6 Determining the PWM Duty Cycle register values

The PWM6DC register is responsible for determining the pulse width and duty cycle. By changing the bit values in the PWM6DC register within a certain range, the values of the pulse width varies and as a result the servo rotates to the appropriate position. To determine when the servo motor rotates to 0°, 90°, and 180°, the value of the bits in the PWM6DC register can be manually changed in a process of trial and error or it can be calculated using the following formula [3]:

$$Pulse\ width = (PWM6DCH{:}PWM6DCL[7{:}6]) \cdot Tosc \cdot (TMR2PrescaleValue) \quad \ldots (2)$$

Table 2: Manually and calculated register values

Pulse width (ms)	Angle (°)	Calculated value (In decimal)	Manually changed value (in decimal)
1	0	15	9
1.5	90	23	25
2	180	31	37

The manually tested values have a more accurate range of movement than compared to the calculated values. Thus, the manually tested values are used for all further calculations and operations.

4.7 Making the servo and joystick work together

Once the PWM and ADC modules are configured and correctly set up, they are required to work in unison to synchronize the movement between the joystick and the servo motor. This can be achieved by moving the value stored in the ADRES register of the ADC to the PWM6DC register. However, moving the ADRES register value straight into the PWM6DC does not have a significant effect on the control that the joystick has over the servo motor. This is due to the ADRESH value being too large in value in comparison to the PWM6DC values that cause the pulse width of the PWM to change.

Thus, the value stored in ADRESH must be scaled down to a smaller value to be within a reasonable range of the manually measured values that cause the servo to rotate. The value stored in ADRESH when the joystick is centered should be scaled so that it is equal to the value stored in PWM6DC when the servo is at 90°. This is achieved by doing the following mathematical operation:

$$PWM6DC\ value = \frac{ADRESH\ value}{4} - 5 \ldots (3)$$

It is ideal to divide by a multiple of two as it makes the division simple to carry out. A logical right shift can be applied to the twice to the ADRESH value to achieve a division by four.

To improve the accuracy of the movement of the servo motor, the value of the ADRESL is moved in to the PWM6DCL register. This results in a greater range of movement for the servo motor.

5. RESULTS AND DISCUSSION

5.1 Final circuit connection

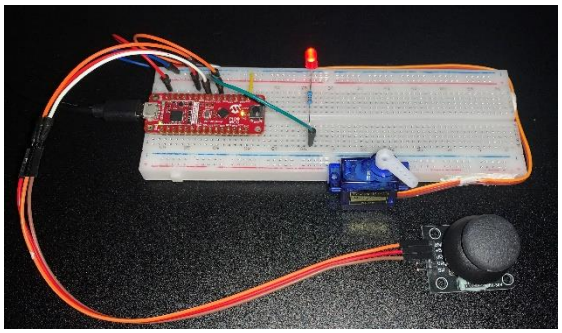

Figure 3: Circuit connection of the components and the microcontroller

5.2 Outcome of the project

The ADC and PWM module are operational as the servo motor is able to move from 0-180° when the joystick is moved, however the positioning of the servo and joystick is not accurate and does not move in the way that is expected. When the joystick is moved along the positive x-axis, the servo moves to 90°. When the joystick is moved above and below the negative x-axis, the servo motor rotates clockwise towards 180°. When the joystick is moved above and below the positive x-axis, the servo moves anti-clockwise towards 0°.

5.3 Recommendations

A way to improve the accuracy of the servo and joystick is to account for the lower bits in the PWM6DCL register when shifting the bits of ADRESH. Since ADRESH is shifted twice, two bits are lost and are not accounted for. These bits need to be moved into bit seven and six of the PWM6DCL register rather than moving the ADRESL value straight into the PWM6DCL register.

6.CONCLUSION

The aim of the project was to create a synchronous system that accurately allows a joystick to control the movement of a servo motor. The ADC and PWM modules of the microcontroller were configured to control the components. The values stored in the ADRES register played an important role in changing the pulse width of the PWM which allowed the joystick to control the servo. However, the established system was not accurate as the movement of the components were not as expected.

REFERENCES

[1] LastMinuteEngineers., *How Servo Motor Works & Interface It With Arduino*. https://lastminuteengineers.com/servo-motor-arduino-tutorial/ , Last accessed 7 May 2022.

[2] LastMinuteEngineers., *How 2-Axis Joystick Works & Interface with Arduino` + Processing.* https://lastminuteengineers.com/joystick-interfacing-arduino-processing/, Last accessed 7 May 2022.

[3] Microchip Technology Inc., 14/20-Pin Full-Featured, Low Pin Count Microcontrollers with XLP, DS50002808C datasheet, 2020.