

# **MPMC PROJECT** **REPORT**

Automatic Toll Gate

26.04.2022

**SUBMITTED BY**

NATASHA DAS

B200264EE

MOGULLAPALLY SUDHEEKSHNA

B200262EE

## Overview

Automation in the transport system is remarkable for the development of smart cities. Keeping that in mind, this project shows the working of automatic toll gates. When any vehicle is sensed, the gate automatically opens reducing the dependence on manual labor.

## Objective

To create an automatic toll gate with an opening and closing delay of 3 - 5 seconds.

## Components Required

1. PIC 18F4550
2. Servo Motor SG90
3. IR sensor
4. Breadboard
5. 4 x 1.5V cells
6. Cell holder
7. Jumper wires
8. PIC-kit3
9. MPLab x IDE 5.1

## Theory

### SERVOMOTOR



The function of the servo motor is to convert the control signal of the controller into the rotational angular displacement or angular velocity of the motor output shaft. Servo motor is used to drive the joints.

0.6ms - -90 degree rotation

1.4ms - 0 degree rotation

2ms - 90 degree rotation

## IR SENSOR



An infrared sensor (IR sensor) is a radiation-sensitive optoelectronic component with a spectral sensitivity in the infrared wavelength range 780 nm. IR sensors are now widely used in motion detectors

Sensing distance can be increased or reduced by rotating the screw clockwise or anticlockwise respectively.

## PIC 18f4550



PIC18F4550 is an 8-bit microcontroller manufactured by Microchip with nano-Watt technology with enhanced flash, USB, and high-performance. It is a 40-pin microcontroller that comes with several features such as memory endurance, self-programmability, extended instruction set, enhanced CCP module, and addressable USART and 10-bit ADC

## Code

```
#include <pic18f4550.h>

#include <stdio.h>

#include <math.h>


#define MINTHR      8000

#define RESOLUTION  488


#define InternalOsc_8MHz  8000000
#define InternalOsc_4MHz  4000000
#define InternalOsc_2MHz  2000000
#define InternalOsc_1MHz  1000000
#define InternalOsc_500KHz 500000
#define InternalOsc_250KHz 250000
#define InternalOsc_125KHz 125000
#define InternalOsc_31KHz  31000


#define Timer2Prescale_1  1
#define Timer2Prescale_4  4
#define Timer2Prescale_16 16

#define mybit PORTBbits.RB4


/* define for setting direction */


void PWM_Init() /* Initialize PWM */
{
    TRISCbits.TRISC2 = 0; /* Set CCP1 pin as output for PWM out */
```

```
CCP1CON = 0x0C;    /* Set PWM mode */
}


int setPeriodTo(unsigned long FPWM)/* Set period */
{
    int clockSelectBits, TimerPrescaleBits;
    int TimerPrescaleValue;
    float period;
    unsigned long FOSC, _resolution = RESOLUTION;

    if (FPWM < MINTHR) {TimerPrescaleBits = 2; TimerPrescaleValue = Timer2Prescale_16;}
    else                {TimerPrescaleBits = 0; TimerPrescaleValue = Timer2Prescale_1;}

    if (FPWM > _resolution)        {clockSelectBits = 7; FOSC = InternalOsc_8MHz;}
    else if (FPWM > (_resolution >>= 1)) {clockSelectBits = 6; FOSC = InternalOsc_4MHz;}
    else if (FPWM > (_resolution >>= 1)) {clockSelectBits = 5; FOSC = InternalOsc_2MHz;}
    else if (FPWM > (_resolution >>= 1)) {clockSelectBits = 4; FOSC = InternalOsc_1MHz;}
    else if (FPWM > (_resolution >>= 1)) {clockSelectBits = 3; FOSC = InternalOsc_500KHz;}
    else if (FPWM > (_resolution >>= 1)) {clockSelectBits = 2; FOSC = InternalOsc_250KHz;}
    else if (FPWM > (_resolution >>= 1)) {clockSelectBits = 1; FOSC = InternalOsc_125KHz;}
    else                            {clockSelectBits = 0; FOSC = InternalOsc_31KHz;}

    period = ((float)FOSC / (4.0 * (float)TimerPrescaleValue * (float)FPWM)) - 1.0;
    period = round(period);

    OSCCON = ((clockSelectBits & 0x07) << 4) | 0x02;
    PR2 = (int)period;
    T2CON = TimerPrescaleBits;
    TMR2 = 0;
    T2CONbits.TMR2ON = 1; /* Turn ON Timer2 */
}
```



```
    return (int)period;
}

void SetDutyCycleTo(float Duty_cycle, int Period)
{
    int PWM10BitValue;


    PWM10BitValue = 4.0 * ((float)Period + 1.0) * (Duty_cycle/100.0);
    CCPR1L = (PWM10BitValue >> 2);
    CCP1CON = ((PWM10BitValue & 0x03) << 4) | 0x0C;
}

void delay(unsigned int val)
{
    unsigned int i,j;
    for(i=0;i<val;i++)
        for(j=0;j<10;j++);
}

int main()
{
    int Period;

    PWM_Init();          /* Initialize PWM */
    Period = setPeriodTo(50); /* 50Hz PWM frequency */
    /* Note that period step size will gradually increase with PWM frequency */

    TRISBbits.TRISB4 = 1 ;
    while(1){
```



```
if(mybit==1){

    delay(3000);

    SetDutyCycleTo(12.0, Period); /* 12% duty cycle */

}

else{

    delay(3000);

    SetDutyCycleTo(7.6, Period); /* 3% duty cycle */

}

}

}
```

## Code Explanation

The above C code controls the positioning of the servo motor based on the feedback received from the IR obstacle sensor.

Internal Clock frequency of 8Mhz has been used. A pulse of on-time of approximately 1.4ms and 2ms has been sent to the CCP1 pin of port C through PWM.

Pulse of frequency of 50 Mhz has been generated and corresponding value has been loaded to the PR2 register.

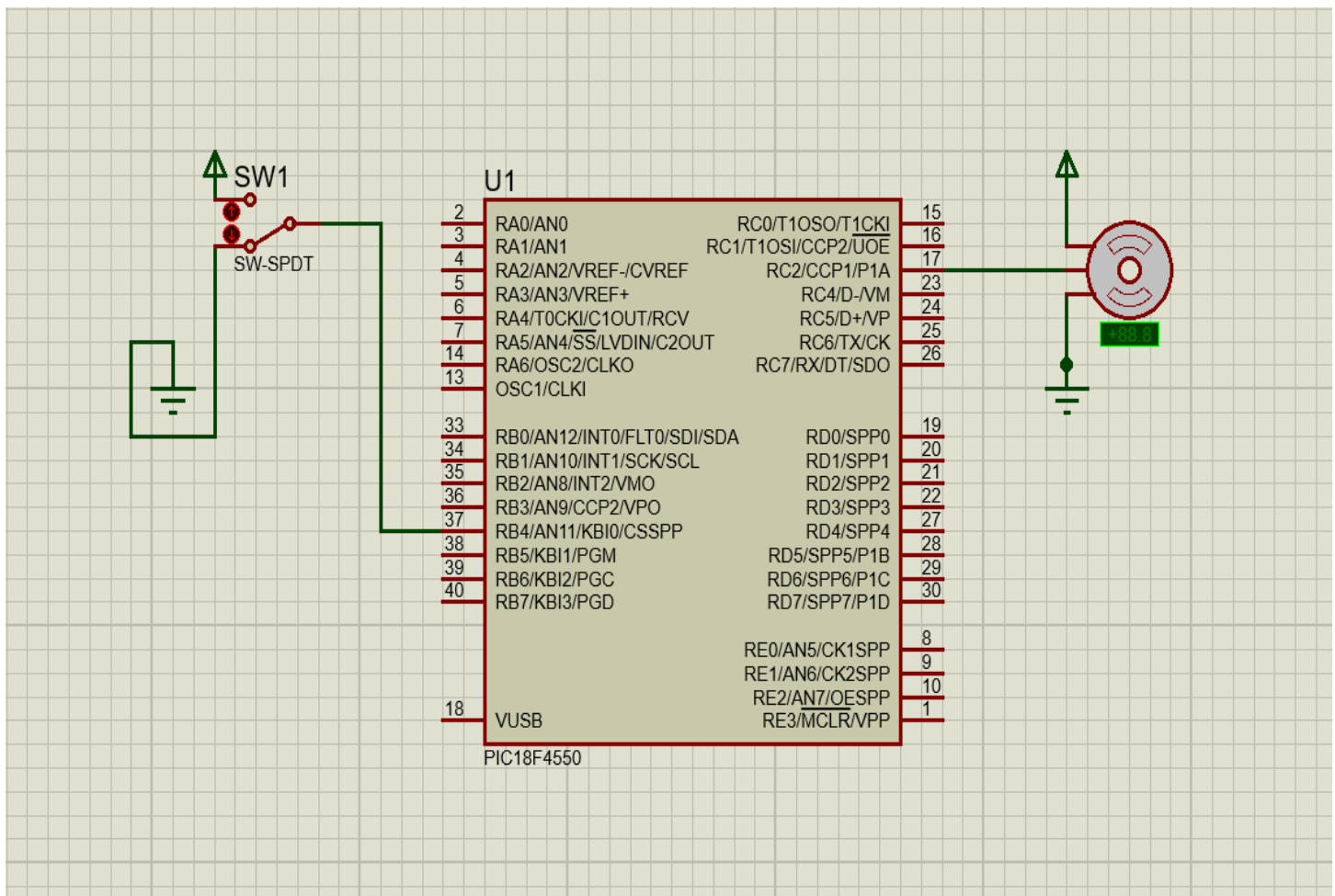
T2CON register has been loaded to make use of timer 2 for the functioning of PWM mode.

When the IR sensor pin is set high, a PWM wave of duty cycle 12% is sent to the servo motor, making it open the gate.

Similarly, when the IR sensor pin is set low, a PWM wave of duty cycle 7% is sent to the servomotor, closing the gate.

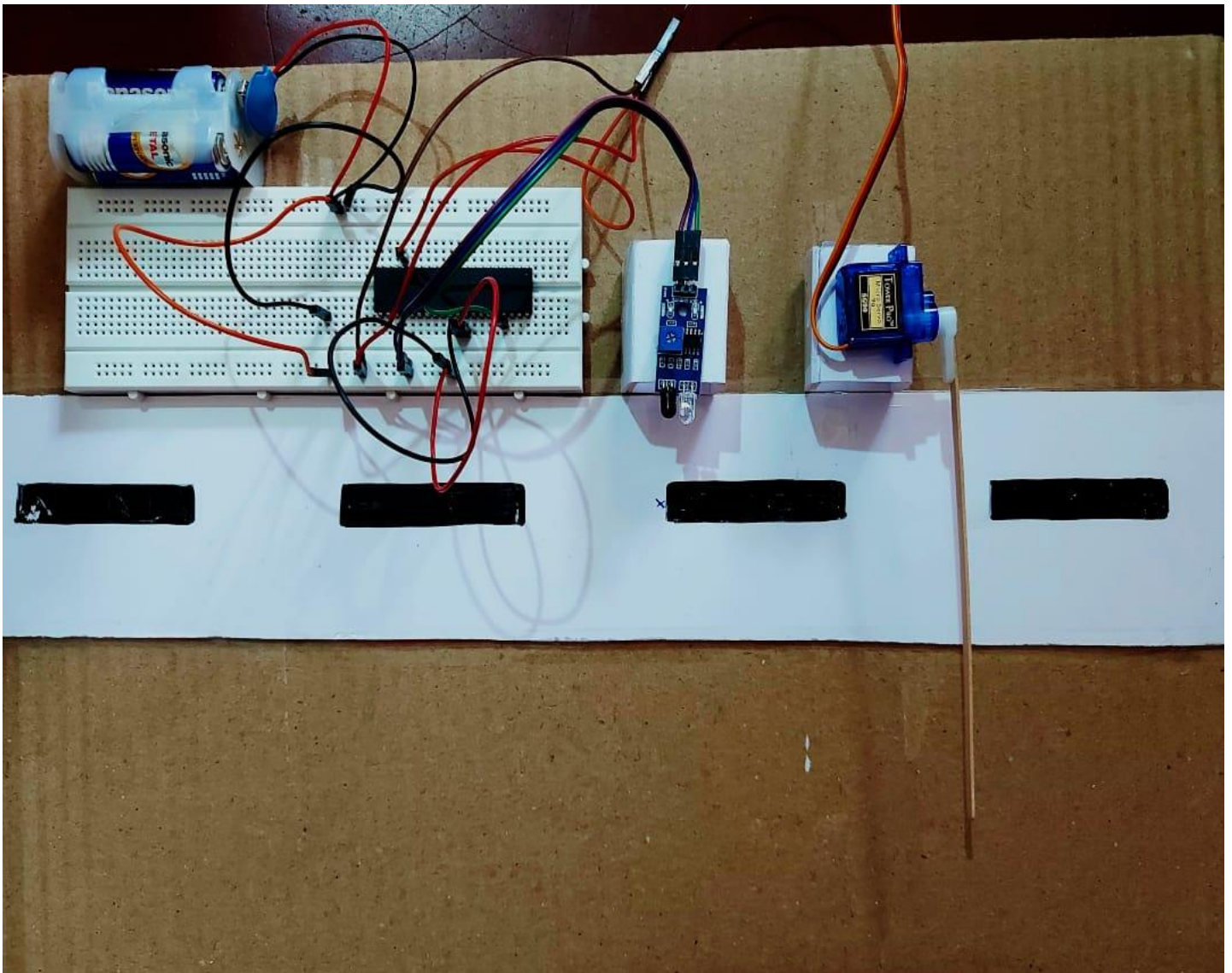
A delay of 3 sec between each action has been generated using the delay function.

## Simulation





## Project Hardware





## **Result**

An automatic toll gate based on IR sensor and the servo motor interfacing through PIC18F4550 has been successfully implemented both in simulation and hardware

## **Inference**

This project has applications in various streams like

- Automatic sensor gates
- Smart Dustbins
- Railway Gates

Therefore, the project is very useful in practical life for multiple purposes.