/\* File: ADtest.c

\* Author: Craig Bacon Created on 02 November 2021

\* Reads the left hand IR distance sensor connected to analogue input AN0(Pin2)

\* If the value of the IR sensor is greater than setpoint\_distance turn on LED3.

\*/

#include <xc.h>

#include <stdio.h>

#include <stdlib.h>

#pragma config OSC = HS

#pragma config WDT = OFF

#pragma config LVP = OFF

#pragma config PWRT = ON

#define LED1 LATBbits.LATB2 //LED1

#define LED2 LATBbits.LATB3 //LED2

#define LED3 LATBbits.LATB4 //LED3

#define LED4 LATBbits.LATB5 //LED4

#define RA4 LATAbits.LATA4

#define RA5 LATAbits.LATA5

#define RB0 LATBbits.LATB0

#define RB1 LATBbits.LATB1

#define \_XTAL\_FREQ 10000000 // define clock frequency for \_\_delay\_10ms()

int setpoint\_distance = 400; //Distance set point

unsigned int readleftADC(void); //Read ADC

unsigned int readrightADC(void); //Read ADC

void wait10ms(int del); //generates a delay in multiples of 10ms

int main(void)

{

int turning = 100; //mark space value for 8 PWM (50% mark space ratio)

int forward = 100;

TRISCbits.RC1=0; //set CCP1(pin13) to an output pin

TRISCbits.RC2=0; //set CCP1(pin13) to an output pin

TRISA = 0b11001111;

TRISB = 0b00000000;

PR2 = 0b11111111 ; //set period of PWM

T2CON = 0b00000111 ; //Timer 2(TMR2) on, Prescaler = 16

CCP2CON = 0b00001100; //0x0c enables PWM module CCP1

CCP1CON = 0b00001100; //0x0c enables PWM module CCP1

CCPR2L = turning; //Load duty cycle into CCP1CON, PWM begins

CCPR1L = turning; //Load duty cycle into CCP1CON, PWM begins

ADCON1=0b00001101; //Set voltage reference and ports A0 and A1 as analogue input

ADCON2 = 0b10000010; // Fosc/32, A/D result right justified

LATB=0; //Turn all Leds off

while(1){

if(readleftADC() >= setpoint\_distance && readrightADC() < setpoint\_distance) //If left hand sensor detects an object

{

LED3=1; //equal or greater than setpoint\_distance

RA4 = 1; //brake

RA5 = 1;

RB0 = 1;

RB1 = 1;

wait10ms(100);

RA4 = 1; //reverse

RA5 = 0;

RB0 = 1;

RB1 = 0;

wait10ms(200);

RB0 = 1; // spinning clockwise

RB1 = 0;

RA4 = 0;

RA5 = 1;

wait10ms(150);

RA4 = 0; //forward

RA5 = 1;

RB0 = 0;

RB1 = 1;

wait10ms(300);

RB0 = 0; // spinning anticlockwise

RB1 = 1;

RA4 = 1;

RA5 = 0;

wait10ms(150);

RA4 = 0; // forward

RA5 = 1;

RB0 = 0;

RB1 = 1;

}

if(readleftADC() < setpoint\_distance) //turn on LED3

{

LED3=0;

}

if(readrightADC() >= setpoint\_distance && readleftADC() < setpoint\_distance) //If left hand sensor detects an object

{

LED4=1; //equal or greater than setpoint\_distance

RA4 = 1; //brake

RA5 = 1;

RB0 = 1;

RB1 = 1;

wait10ms(100);

RA4 = 1; //reverse

RA5 = 0;

RB0 = 1;

RB1 = 0;

wait10ms(200);

RB0 = 0; // spinning anticlockwise

RB1 = 1;

RA4 = 1;

RA5 = 0;

wait10ms(150);

RA4 = 0; //forward

RA5 = 1;

RB0 = 0;

RB1 = 1;

wait10ms(300);

RB0 = 1; // spinning clockwise

RB1 = 0;

RA4 = 0;

RA5 = 1;

wait10ms(150);

RA4 = 0; // forward

RA5 = 1;

RB0 = 0;

RB1 = 1;

}

if(readrightADC() < setpoint\_distance) //turn on LED3

{

LED4=0;

}

if(readrightADC() >= setpoint\_distance && readleftADC() >= setpoint\_distance) //If left hand sensor detects an object

{

LED4=1; //equal or greater than setpoint\_distance

RA4 = 1; //brake

RA5 = 1;

RB0 = 1;

RB1 = 1;

wait10ms(100);

RA4 = 1; //reverse

RA5 = 0;

RB0 = 1;

RB1 = 0;

wait10ms(200);

RB0 = 0; // spinning anticlockwise

RB1 = 1;

RA4 = 1;

RA5 = 0;

wait10ms(150);

RA4 = 0; //forward

RA5 = 1;

RB0 = 0;

RB1 = 1;

wait10ms(300);

RB0 = 1; // spinning clockwise

RB1 = 0;

RA4 = 0;

RA5 = 1;

wait10ms(150);

RA4 = 0; // forward

RA5 = 1;

RB0 = 0;

RB1 = 1;

}

}

}

unsigned int readleftADC(void) {

ADCON0 = 0b00000011; //select A/D channel AN0,start conversion

while (ADCON0bits.GO); //do nothing while conversion in progress

return ((ADRESH << 8) + ADRESL); //Combines high and low A/D bytes into one

} // value and returns Result (A/D value 0-1023)

unsigned int readrightADC(void) {

ADCON0 = 0b00000111; //select A/D channel AN0,start conversion

while (ADCON0bits.GO); //do nothing while conversion in progress

return ((ADRESH << 8) + ADRESL); //Combines high and low A/D bytes into one

} // value and returns Result (A/D value 0-1023)

void wait10ms(int del){ //delay function

unsigned char c;

for(c=0;c<del;c++)

\_\_delay\_ms(10);

return;

}