Continuous Assessment Cover Sheet

Faculty of Engineering

Module Details			
Module Code	**EC2132**	**Module Title**	**Microcomputers**
Program: SLIIT		**Course: BSc in Electrical and Electronic Engineering**	
Stream: EEE			
Assessment details			
Title	**Laboratory 04**	**Group assignment**	**Yes**
		If yes, Group No.	**18**
Instructor		**Date of Performance**	
Due date	**19.09.2022**	**Date submitted**	**18.08.2022**

Student statement and signature

By this declaration, I/we confirm my/our understanding and acceptance that the work reported in this report is my/our own work. I/we also understand the consequences of engaging in plagiarism or copying others work without proper citation. Any material used in this work (whether from published sources, the internet or elsewhere) have been fully acknowledged and referenced and are without fabrication or falsification of data.

[Copying or plagiarism will result in a “0” mark for the continuous assessment and “F” for the module after an investigation on academic misconduct.

All academic misconduct is considered seriously and defined as dishonest and in direct opposition to the values of a learning community. Misconduct may result in penalties from failure to exclusion from the campus.
Further help and guidance on how to avoid academic misconduct can be obtained from your academic advisor/tutor]

By this declaration, I/we confirm my understanding and acceptance that-

- **I/we have adhered to relevant ethical guidelines and procedures in the completion of the assignment.**
- **I/we have not allowed another student to have access to or copy from this work.**
- **This work has not been submitted previously.**

[The Institute may request an electronic copy of this work for submission to the Plagiarism detection facility (TURNITIN). You must make sure that an electronic copy of your work is available in these circumstances]

Details of the students submitting the assignment		**Signature**
ID Number	**Name (As per the institute records)**	
EN21474682	**Keerthirathne A.H.R.**	

EN21497384	Perera M.T.S.M.	
EN2121479182	Somarathne S.S.	Somarathne..

OFFICE USE ONLY

Receiving Officer (seal, signature, date)	**Specific comments about the work (including overall comments and guidelines for improvement)**		
	Tutor:	**Signature:**	**Date:**
	Marks:	[All marks are subject to external moderation and approval of board of examinations]	

SRI LANKA INSTITUTE OF INFORMATION TECHNOLOGY

Microcomputers (EC2132)
Laboratory 04

Group members:

Somarathne S.S. - EN21479182

Keerthirathne A.H.R. – EN21474682

Perera M.T.S.M. – EN21497384

Title: Laboratory 04

Objectives

- To develop a small water level control system in a water tank using the knowledge of PIC16F877A interrupts and other programming techniques from all the labs we have done throughout the course.

Introduction

A water tank level control system is a system that monitors water levels for us and automatically controls pumps and other devices to reliably control water levels in water storage tanks and other applications. PIC16F877A microcontroller, interrupts in PIC microcontroller, microcontroller programming in C language and other hardware methods studied so far were used to develop this small water level control system.

1. PIC16F877A Microcontroller

The PIC16F877a is a 40-pin PIC Microcontroller manufactured by Microchip, designed using the Reduced Instruction Set Computing (RISC) architecture, and used in Embedded Projects. PIC16 microcontrollers family.

Figure 1 – PIC16F877A microcontroller

The pin configuration of the PIC16F877A microcontroller is shown as below.

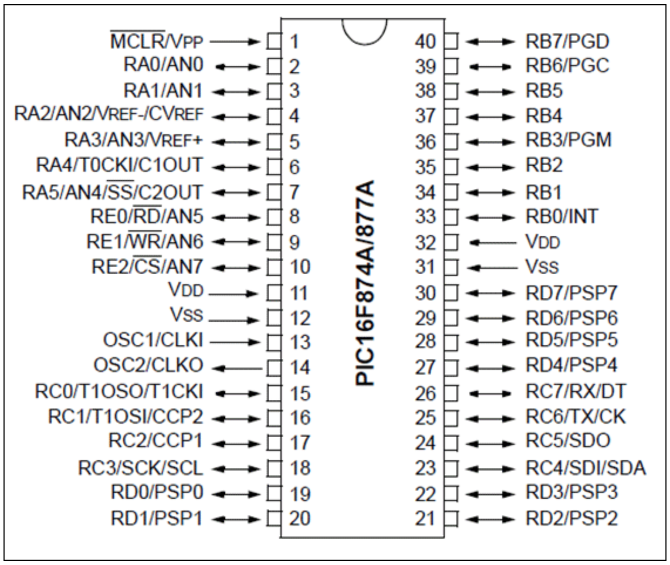


Figure 2 – The pin configuration of the PIC16F877A microcontroller

2. L293D Motor driver

The L293D is a 16-pin motor driver IC that can simultaneously operate two DC motors in either direction. The L293D can deliver bidirectional drive currents up to 600 mA (per channel) at voltages ranging from 4.5 V to 36 V (at pin 8!). It can be used to manage small dc motors.

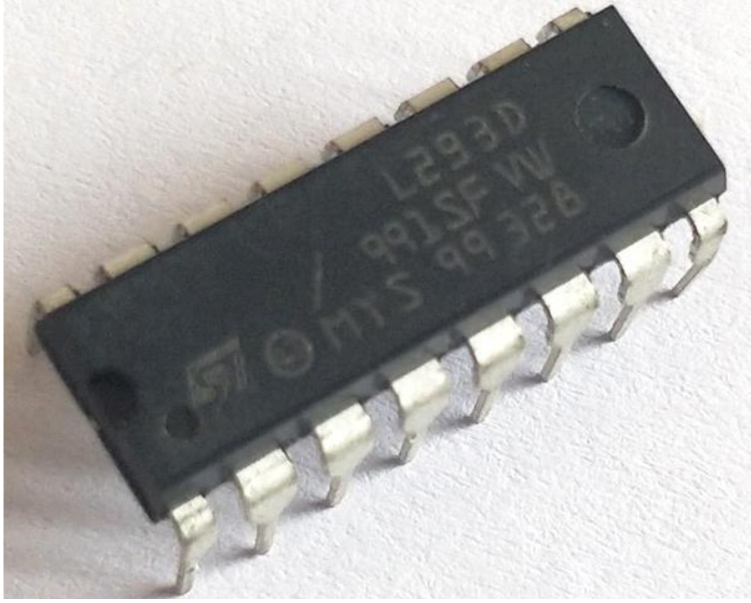


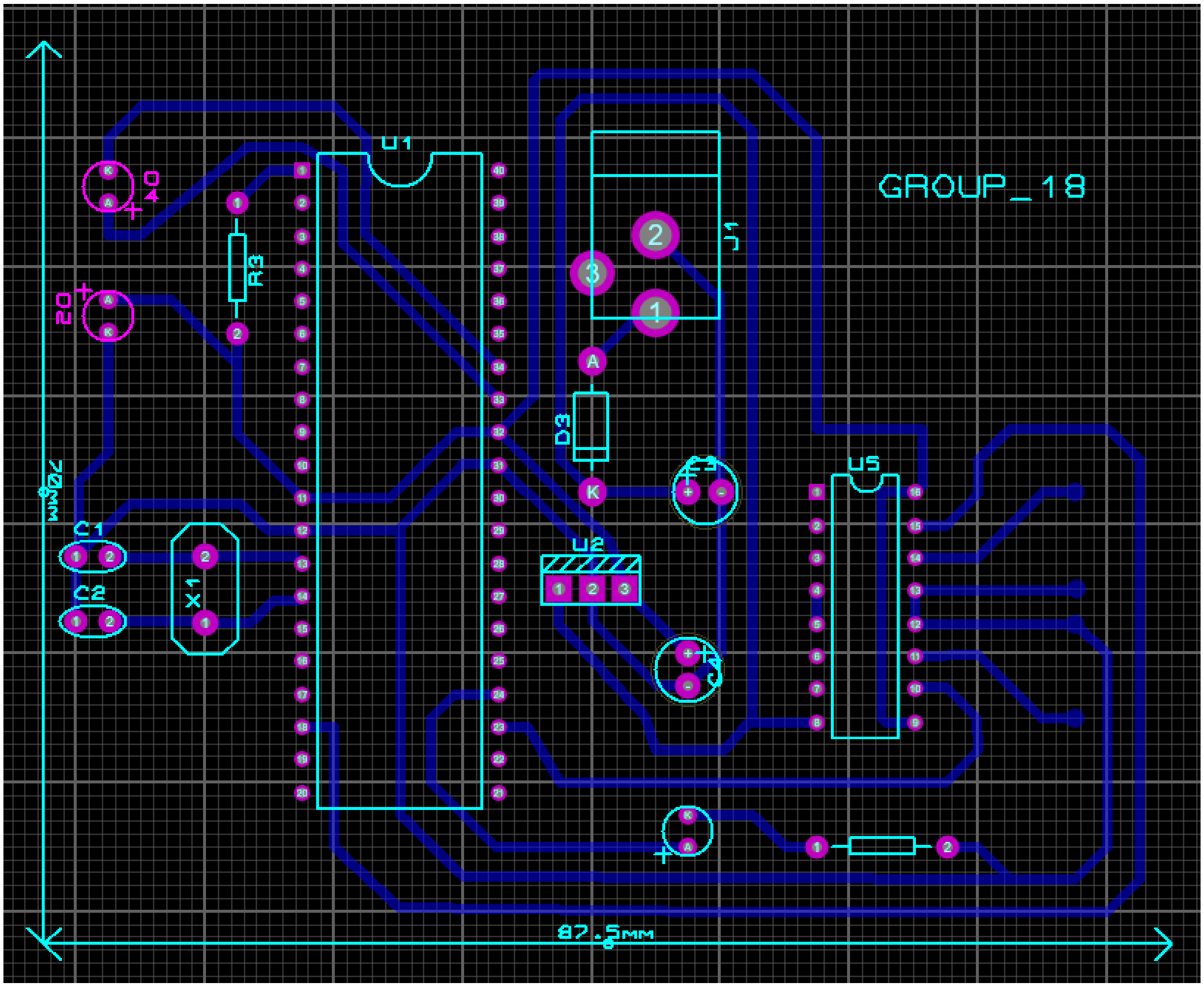
Figure 3 – L293D Motor driver

Apparatus

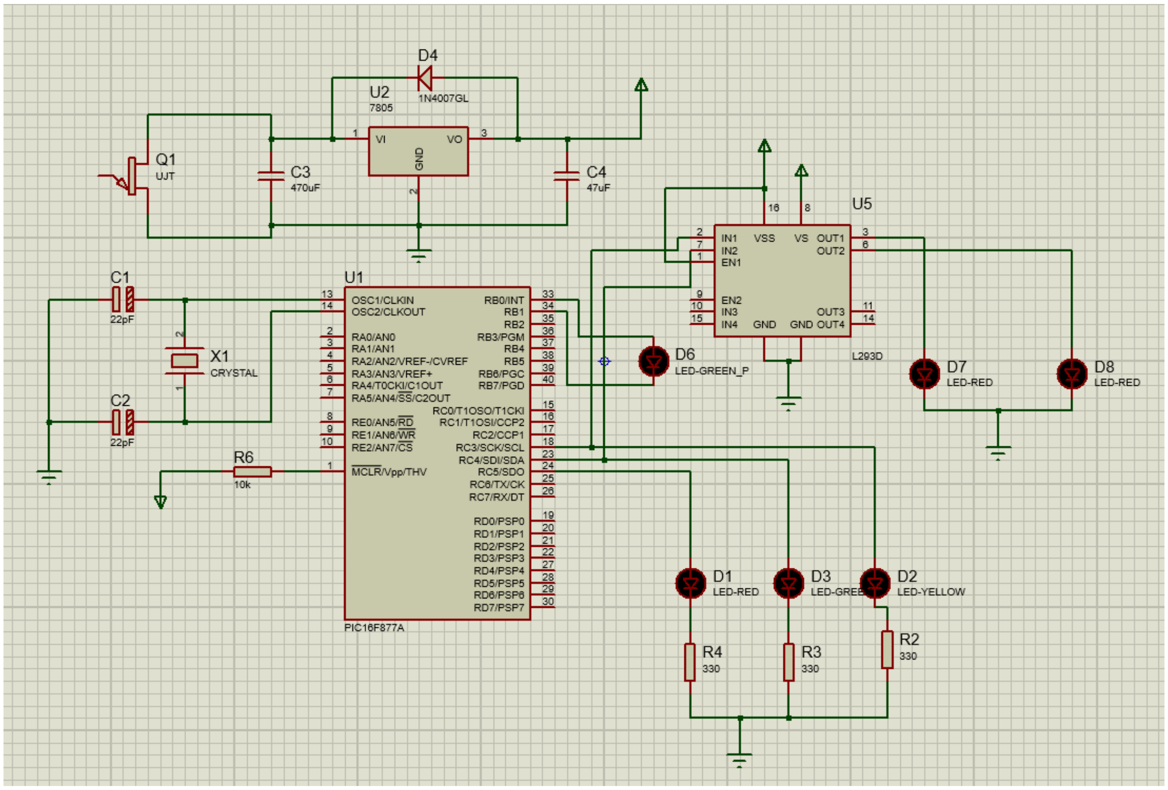
- PIC16F877A Microcontroller
- Two DC Motors
- Crystal oscillator
- Resistors
- Capacitors (47uF & 470uF)
- Diode 1N4007RLG
- LED’s
- L293D Motor driver
- Ultrasonic sensor
- Power input (12V DC)

Lab work

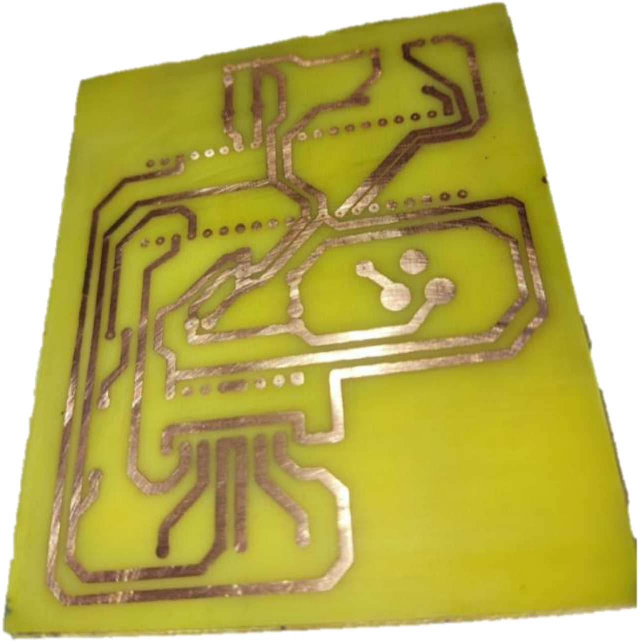
- The schematic View of PCB :-



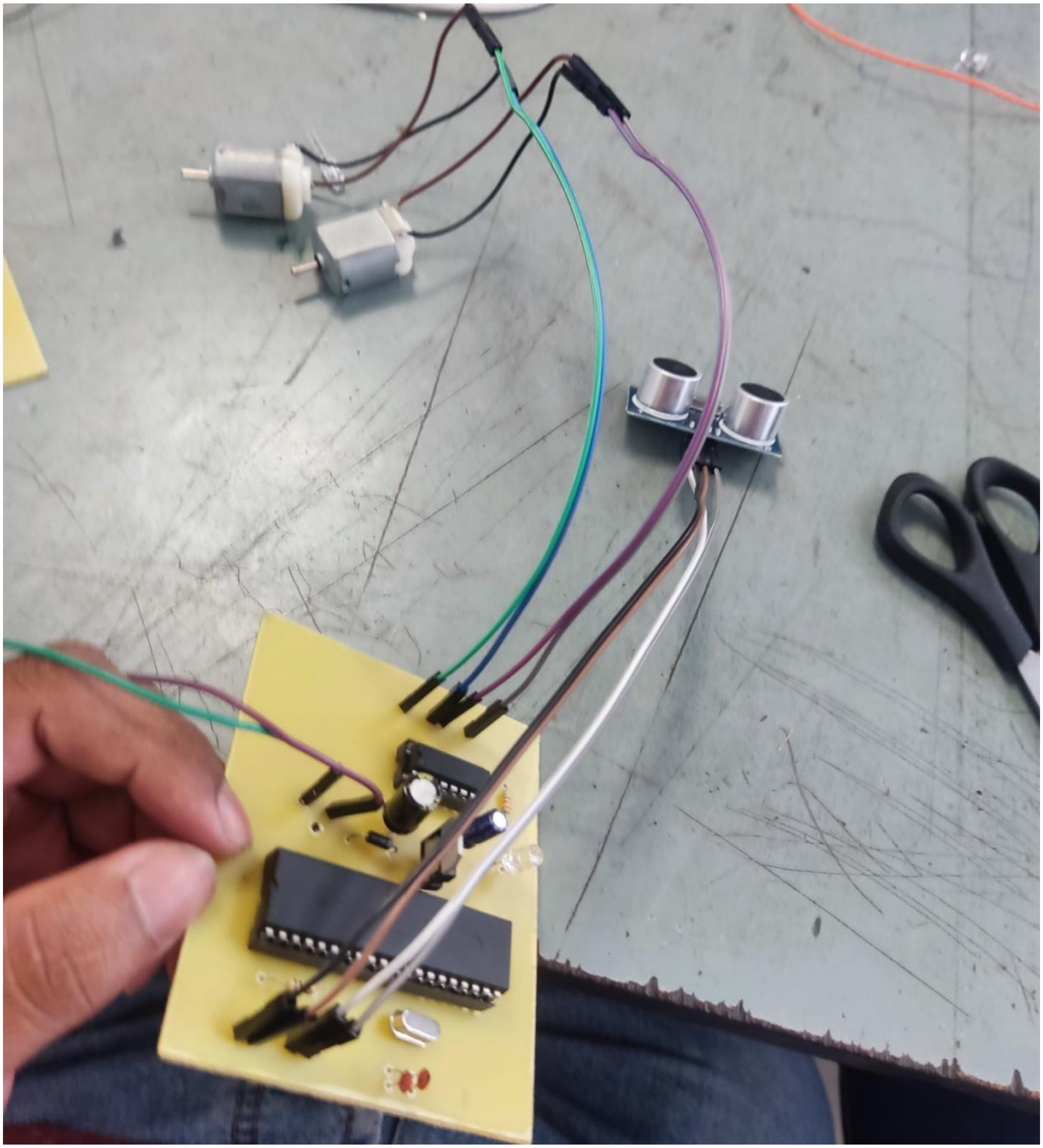
- The schematic View of the system :-



- Created PCB Design :-



- The real implementation of the system :-



- The code :-

```
// CONFIG
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial Programming
Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM
code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection
off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection
off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#define _XTAL_FREQ 20000000
#define trigger RB1
#define echo RB0
//#define trigger1 RB3
//#define echo1 RB0

void __interrupt() timer_isr (void){

  if(TMR1IF==1){
      RC5=1;
      RC3=0;
      RC4=0;
      __delay_ms(500);
      RC5=0;
      TMR1IF=0;
      }
  }

void main (void){
  TRISB0=1;    // INTERRUPT PIN ECHO
  TRISB1=0;    //TRIGER PIN
  // TRISB5=1;   // ECHO PIN
  // TRISB3=0;   //SENSER 2 TRIGGER
```

```
   // TRISC0=1;   //c echo
   // TRISC1=1;

   TRISC3=0;  //LED OUTPUT PIN
   TRISC4=0;  //led
   TRISC5=0;   //interrupt led
   T1CON=0X20;  // 4- PRES-SCALAR AND INTERNAL CLOCK
   RC3=0;
   RC4=0;
   RC5=0;
   //  INTF=0;     //INTF VALUE SET TO 0

   TMR1IF=0;
   TMR1IE=1;      //Enable timer interrupt bit in PIE1 register
   GIE=1;   //GLOBAL INTERRUPT ENABLE BIT=1(ACCESS TO INTERRUPT)
   PEIE=1;  //PERIPERAL INTERRUPT ENABLE BIT=1(CONTROLL THE EXTERNAL
DEVICES USING INTERRUPT)

   T1SYNC=0;
   //INTE=1;  //INTE SET TO 1(CONVERT PUSH BOTTON INPUT AS INTERRUPT INPUT)

   int time_taken;
   int distance;
  // int time_taken1;
  // int distance1;

  // char t1,t2,t3,t4,t5;
  //  char d1,d2,d3;

   while(1){

       TMR1H = 0;   //CLEAR THE TIMER BIT
       TMR1L = 0;

       trigger = 1;
       __delay_ms(10);
       trigger = 0;

       while(echo==0);

       TMR1ON = 1;

       while(echo==1);
```

```
     TMR1ON = 0;

     time_taken = (TMR1L | (TMR1H<<8));
     distance = (0.0272*time_taken)/2;

     time_taken = time_taken*0.8;

    if(distance<20 && 10<distance ){
       RC3=1;

    }
    else{
       RC3=0;
    }

    if(distance<=10){
       RC4=1;
    }
    else{
       RC4=0;

    }

    if(distance<30 && 20<=distance){
       TMR1IF=1;
      // RC5=1;
      // __delay_ms(500);
      // RC5=0
    }
  }


  return;
}
```

- Link for the Demonstration video :-

https://drive.google.com/file/d/1ls6Hii5qiS-6J8QX9q-l2Uq5Ku6RQprG/view