INTERNATIONAL UNIVERSITY

BIOMEDICAL ENGINEERING

**MICRO\_ELECTRONIC DEVICE**

**BM062IU**

FINAL REPORT

GARDEN AUTOMATIC-WATERRING

MICROPROCESSING SYSTEM

**Submitted by**

Võ Quang Trấn - BEBEIU17008

Nguyễn Hoàng Tân - BEBEIU17032

Đinh Hoàng Sáng - BEBEIU17022

Date Submitted: 21/07/2020

Date Performed: 13/07/2020

Lab Section: Final

Course Instructor: M.ENG VO MINH THANH

**GRADING GUIDELINE FOR LAB REPORT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Content** |  | **Score** | **Comment** |
| 1 | **Format (max 9%)** | |  |  |
| * Font type | Yes No |  |
| * Font size | Yes No |  |
| * Lab title | Yes No |  |
| * Page number | Yes No |  |
| * Table of contents | Yes No |  |
| * Header/Footer | Yes No |  |
| * List of figures (if exists) | Yes No |  |
| * List of tables (if exists) | Yes No |  |
| * Lab report structure | Yes No |  |
| 2 | **English Grammar and Spelling (max 6%)** | |  |  |
| * Grammar | Yes No |  |
| * Spelling | Yes No |  |
| 3 | **Data and Result Analysis (max 85%)** | |  |  |
| **Total Score** | |  | |  |

Signature:

Date:

**Table of Contents**

List of Figures ........................................................................................….….............................. 3

List of Tables .................................................................................................……....................... 3

Introduction .................................................................................................……....................... 4

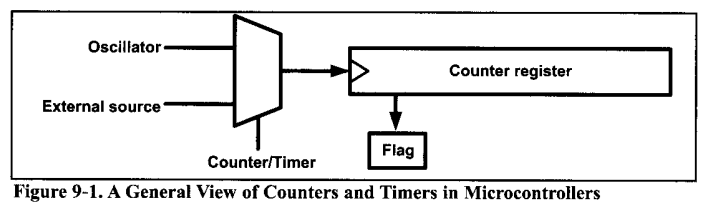
Theoretical Background .............................................................................................…….......... 4

Experimental Procedure.............................................................................................……........... 5

**Theoretical Background**

**1/Timer**

Many applications need to count an event or generate time delays. So, there are counter registers in microcontrollers for this purpose. See Figure 9-1. When we want to count an event, we connect the external event source to the clock pin of the counter register. Then, when an event occurs externally, the content of the counter is incremented; in this way, the content of the counter represents how many times an event has occurred. When we want to generate time delays, we con­nect the oscillator to the clock pin of the counter. So, when the oscillator ticks, the content of the counter is incremented. As a result, the content of the counter reg­ister represents how many ticks have occurred from the time we have cleared the counter. Since the speed of the oscillator in a microcontroller is known, we can cal­culate the tick period, and from the content of the counter register we will know how much time has elapsed.



So, one way to generate a time delay is to clear the counter at the start time and wait until the counter reaches a certain number For example, consider a microcontroller with an oscillator with frequency of 1 MHz; in the microcon­troller, the content of the counter register increments once per microsecond. So, if we want a time delay of 100 microseconds, we should clear the counter and wait until it becomes equal to 100.

In the microcontrollers, there is a flag for each of the counters. The flag is set when the counter overflows, and it is cleared by software. The second method to generate a time delay is to load the counter register and wait until the counter overflows and the flag is set. For example, in a microcontroller with a frequency of 1 MHz, with an 8-bit counter register, if we want a time delay of 3 microsec­onds, we can load the counter register with $FD and wait until the flag is set after 3 ticks. After the first tick, the content of the register increments to $FE; after the second tick, it becomes $FF; and after the third tick, it overflows (the content of the register becomes $00) and the flag is set.

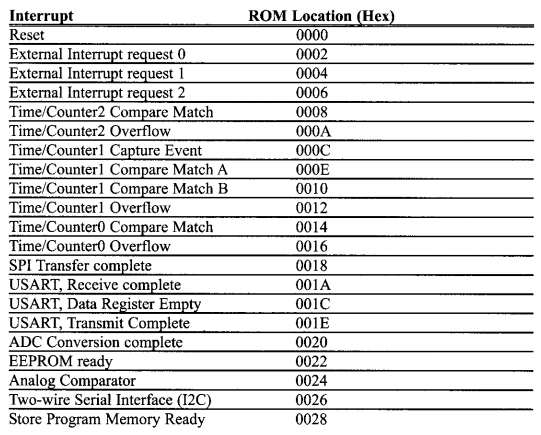
The AVR has one to six timers depending on the family member. They are referred to as Timers 0, 1,2, 3, 4, and 5. They can be used as timers to generate a time delay or as counters to count events happening outside the microcontroller.

In the AVR some of the timers/counters are 8-bit and some are 16-bit. In ATmega32, there are three timers: TimerO, Timerl, and Timer2. TimerO and Timer2 are 8-bit, while Timerl is 16-bit. In this chapter we cover TimerO and Timer2 as 8-bit timers, and Timerl as a 16-bit timer.

If you learn to use the timers of ATmega32, you can easily use the timers of other AVRs. You can use the 8-bit timers like the TimerO of ATmega32 and the 16-bit timers like the Timerl of ATmega32.

**2/Interrupt**

A single microcontroller can serve several devices. There are two methods by which devices receive service from the microcontroller: interrupts or polling. In the interrupt method, whenever any device needs the microcontroller’s service, the device notifies it by sending an interrupt signal. Upon receiving an interrupt signal, the microcontroller stops whatever it is doing and serves the device. The program associated with the interrupt is called the interrupt service routine (ISR) or interrupt handler. In polling, the microcontroller continuously monitors the sta­tus of a given device; when the status condition is met, it performs the service. After that, it moves on to monitor the next device until each one is serviced. Although polling can monitor the status of several devices and serve each of them as certain conditions are met, it is not an efficient use of the microcontroller. The advantage of interrupts is that the microcontroller can serve many devices (not all at the same time, of course); each device can get the attention of the microcon­troller based on the priority assigned to it. The polling method cannot assign pri­ority because it checks all devices in a round-robin fashion. More importantly, in the interrupt method the microcontroller can also ignore (mask) a device request for service. This also is not possible with the polling method. The most important reason that the interrupt method is preferable is that the polling method wastes much of the microcontroller’s time by polling devices that do not need service. So interrupts are used to avoid tying down the microcontroller. For example, in dis­cussing timers in Chapter 9 we used the bit test instruction “SBRS R2 0, TOV0” and waited until the timer rolled over, and while we were waiting we could not do anything else. That is a waste of microcontroller time that could have been used to perform some useful tasks. In the case of the timer, if we use the interrupt method, the microcontroller can go about doing other tasks, and when the TOVO flag is raised, the timer will interrupt the microcontroller in whatever it is doing.



**3/LCD**

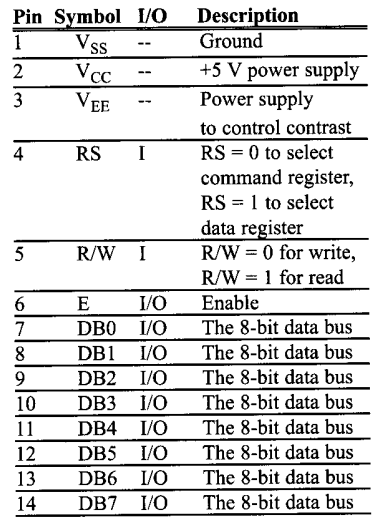
In recent years the LCD is finding widespread use replacing LEDs (seven- segment LEDs or other multisegment LEDs). This is due to the following reasons:

1. The declining prices of LCDs.

2. The ability to display numbers, characters, and graphics. This is in contrast to LEDs, which are limited to numbers and a few characters.

3. Incorporation of a refreshing controller into the LCD, thereby relieving the CPU of the task of refreshing the LCD. In contrast, the LED must be refreshed by the CPU (or in some other way) to keep displaying the data.

4. Ease of programming for characters and graphics.



**Experimental Procedure:**

**Code:**

/\* Võ Quang Trấn, Nguyễn Hoàng Tân, Đinh Hoàng Sáng \*/

/\* Atmega32-finalproject-Water-pumping micro-control system in AVR programer-C language \*/

/\* FINALPROJECT\_AUTOMATICAL\_DREAMGARDEN

\*

\*

\* Created: 8/9/2020

\* Author :

Instructor: M.Eng Vo Minh Thanh

\*/

#define F\_CPU 8000000UL

#include <stdio.h>

#include "lcd.h"

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdbool.h>

#include <util/delay.h>

#include "adc.h"

#include "stdlib.h"

#include "keypad.h"

#include "rtc.h"

#define SS 4 // Slave Select is Bit No.4

#define MOSI 5 // Master Out Slave In is Bit No.5

#define MISO 6 // Master In Slave Out is Bit No.6

#define SCK 7 // Shift Clock is Bit No.7

char Shownhietdo[2],Showdoam[2],Shownguoi[3],Showkey[1]; //declare buffer for displaying temperature(temperature sensor), moisture(from SPI TC72 sensor), people (IR sensor)

int cout2=0,cout1=0,stt=1,count=0,cout3=0; //declare counting variables

uint8\_t key,minute,hour,alarmode=0; //declare variables for RTC

double Alarm=0,Time=0; //declare variables for Autonomous Pumping Mode

static volatile uint8\_t LCDtt=1; //declare variables to keep track of LCD screen

static volatile int doamsetup=50,tempsetup=50; //declare variables to input temperature and moisture from PC through UART

volatile char ch,rx\_flag=0; //declare UART-receiving flag

rtc\_t rtc;

bool Day; //declare this variable to tract daytime/ nighttime

void main()

{

DDRC &= ~(1<<PC2); //Config PC2 input - connected to track number of visitors button

DDRC &= ~(1<<PC3); //Config PC3 input - connected to go back to main menu button

DDRC &= ~(1<<PC4); //Config PC4 input - connected to set up button

PORTC |= (1<<PC2)|(1<<PC3)|(1<<PC4); //Use pull-up resistor for PC2, PC3, PC4

DDRA |= (1<<PA4)|(1<<PA5); //Config PA4, PA5 output

//PA4 - watering pump ; PA5 - mist maker pump

LCDInit(LS\_NONE); //Initialize LCD, no-cursor mode

initADC(); //Initialize ADC

SPI\_MasterInit(); //Initialize SPI, master mode

TC72\_init(); //Initialize TC72

RTC\_Init(); //Initialize RTC

LCDClear(); //Clear LCD display

KEYPAD\_Init(PA\_6,PA\_3,PC\_5,PC\_6,PB\_0,PB\_1,PB\_3,PA\_7); //Initialize keypad at pin PA6, PA3, PC5, PC6, PB0, PB1, PB3, PA7

//Configure timer

TCNT0=178; //100ms

TCCR0= (1<<CS00) | (1<<CS02); // prescaler 1024

TIMSK = (1<<TOIE0); //Enable Timer0 Overflow Interrupt on Timer Interrupt Mask resistor

//Congigure external interrupt

MCUCSR = (1<<ISC2); //Config external hardware interrupt of INT2 is raising edge triggered.

GICR = (1<<INT2); //Enable external hardware interrupt 2

//Configure UART

UCSRB= 0x98; // Enable Receiver,Transmitter,Receiver interrupt(RXIE)

UCSRC= 0x86; // Asynchronous mode 8-bit data and 1-stop bit

UCSRA= 0x00; // Normal Baud rate(no doubling), Single processor commn

UBRRL= 51; // Set baud rate 9600

UBRRH= 0; //

sei (); // enable global interrupts

while (1)

{

if (ReadADC(1)/10>50) //check daytime/ nighttime

{

Day=true; //on daytime

GICR &= ~(1<<INT2); //Disable External Hardware Interrupt INT2, deactivate IR sensor, do not count people

cout1=0; // reset counter

}

else

{

Day=false; //on nighttime

GICR = (1<<INT2); //Enable External Hardware Interrupt INT2, activate IR sensor, count people

cout2=cout1; // save the visitor value of the previous night (the initial people counter = 0 when it turn to daytime)

}

minute = (uint16\_t)rtc.min; //get current minute from real time clock

minute = minute - checknumber(minute); //correct minute

hour= (uint16\_t)rtc.hour; ////get current hour from real time clock

hour= hour - checknumber(hour); //correct hour

Time = (minute)+(hour)\*60; //compute time for comparison (alarm mode)

if ((alarmode==1)&&(Time==Alarm)) //if in alarm mode is turned on and setup time = alarm.

{

PORTA |= (1<<PA4); //activate water pump

}

if (count == 10) //

{

count=0; //Interrupt overflow each 1s

if (ReadADC(0)/10<=doamsetup) //compare setup moisture with measured moisture

{

PORTA |= (1<<PA4); //activate moisture

}

else {PORTA &= ~(1<<PA4);} ////deactivate moisture

if (ReadADC(2)/2-1>=tempsetup) //compare setup temperature with measured temperature

{

PORTA |= (1<<PA5); //activate temperature

}

else {PORTA &= ~(1<<PA5);} ////deactivate temperature

}

switch (LCDtt)

{

case 1:

LCDchinh(); //LCD display main screen

break;

case 2:

LCDsntc(); //LCD display people monitor

break;

case 3:

LCDsetup(); //LCD display set up temperature & moisture form UART

break;

case 4:

LCDhengio(); //LCD display timer-pump set up from keyboard

break;

}

if (PINC==(PINC&~(1<<PC2))) //check PC2 status - people-monitoring button pressed

{

LCDtt=2; //change the screen tracker to 2

\_delay\_ms(200);

}

if (PINC==(PINC&~(1<<PC3))) //check PC3 status - main screen button pressed

{

LCDtt=1; //change the screen tracker to 1

\_delay\_ms(200);

}

if (PINC==(PINC&~(1<<PC4))) //check PC4 status - setup button

{

cout3++; //change state of setup button

if (cout3==1)//press 1 time

{

LCDtt=3;//go to temperature & moisture screen

}

if (cout3==2)//press 2 time

{

LCDtt=4;//go to timer-pump set up screen

cout3=0;//

}

\_delay\_ms(300);//

}

}

}

void LCDsntc(void)

{

LCDClear();

LCDGotoXY(0,0);

LCDWriteString("Dem qua co:");

itoa(cout2,Shownguoi,10); //convert number of people (from type int to string) to display

LCDGotoXY(0,1);

LCDWriteString(Shownguoi);

LCDWriteString(" truy cap");

\_delay\_ms(100);

}

void LCDchinh(void){

LCDClear();

LCDGotoXY(0,0);

LCDWriteString("Nhietdo:");

itoa(gettemperature(),Shownhietdo,10); //convert setup temperature (from type int to string) to display

//itoa(tempsetup,Shownhietdo,10);

LCDWriteString(Shownhietdo);

LCDGotoXY(0,1);

LCDWriteString("Soil:");

//itoa(ReadADC(0)/(ReadADC(2)/2-1),Showdoam,10);

itoa(doamsetup,Showdoam,10); //convert setup moisture (from type int to string) to display

LCDWriteString(Showdoam);

LCDWriteString(" %");

\_delay\_ms(100);

}

void SPI\_MasterInit(void){

DDRB |= (1<<MOSI) | (1<<SCK) | (1<<SS); // config MOSI, SCK, SS output

DDRB &= ~(1<<MISO); //config MISO input

SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR0); // enable SPI, master mode, Shift Clock = CLK /16

PORTB &= ~(1<<SS); //// Enable Slave Select Pin

}

void TC72\_init(void){

PORTB |= (1<<SS); // initializing the packet by pulling SS low

SPI\_write(0x80); //write address to control register

SPI\_write(0x10); //write data to control register: continuous temperature conversion

PORTB &= ~(1<<SS); // terminate the packet by pulling SS high

}

float toFloat(signed int tempr) //get temperature in binary type and convert to float type

{

float result = (float)(tempr>>8);//

char count = tempr & 0x00C0; //

if(count)

{

count = count >> 6; //

result = result + (count \* 0.25);//

}

return (result);

}

double TC72\_ReadTempr() //Read temperature

{

int temprMSB,temprLSB;

PORTB |= (1<<SS); // initializing the packet by pulling SS low

SPI\_write(0x02); //read address command to MSB temperature resister

temprMSB = SPI\_read();

temprLSB = SPI\_read();

PORTB &= ~(1<<SS); // terminate the packet by pulling SS high

return ( (temprMSB<<8) + temprLSB ); //return in 16 bits datatype

}

void SPI\_write(unsigned char cData){

SPDR = cData; // Start transmission

while(!(SPSR & (1<<SPIF))); // Wait for transmission complete

char clear = SPDR; // return the received data

}

int SPI\_read(){

SPDR = 0; // Start transmission of nothing

while(!(SPSR & (1<<SPIF))); // Wait for transmission complete

return SPDR; //// return the received data

}

int gettemperature(){

int temp=toFloat(TC72\_ReadTempr())\*2;

PORTB &= ~(1<<SS); //terminate the packet by pulling SS high

return temp;

}

ISR (INT2\_vect){ // ISR for external interrupt 2

cout1++; // toggle PORTA.2

}

void tx\_char()

{

while((UCSRA & 0x60)==0); // Wait till Transmitter(UDR) register becomes Empty

UDR =ch; // Load the data to be transmitted

}

ISR (USART\_RXC\_vect)

{

ch=UDR; // copy the received data into ch

rx\_flag=1; //flag is set to indicate a char is received

}

void LCDsetup(){

\_delay\_ms(200);

if (stt==1)

{

LCDClear();

LCDGotoXY(0,0);

LCDWriteString("Temperature?");

if(rx\_flag==1) //receive data complete

{

//tx\_char();

tempsetup=ch; //get temperature form UART PC input

rx\_flag=0;

stt++;

}

}

if (stt==2)

{

LCDClear();

LCDGotoXY(0,0);

LCDWriteString("Do am?");

if(rx\_flag==1)

{

//tx\_char();

doamsetup=ch; //get moisture form UART PC input

rx\_flag=0;

stt=1;

LCDtt=1; //back to main screen

}

}

}

ISR(TIMER0\_OVF\_vect) // Interrupt Overflow each 1s

{

count++;

TCNT0=178; //100ms

RTC\_GetDateTime(&rtc); //get current date and time

}

void LCDhengio(){

while (1)

{

LCDClear();

LCDGotoXY(0,0);

LCDWriteString("ALARM:"); //Write Alarm on the first line

if (alarmode==1) //Consider Alarm mode

{

LCDWriteString(" ON"); //Display alarm state: on, on the same line

\_delay\_ms(50);

}

if (alarmode==0)

{

LCDWriteString(" OFF"); //Display alarm state: off, on the same line

\_delay\_ms(50);

}

if (PINC==(PINC&~(1<<PC3))) //if PC3 pressed

{

alarmode=1; //Alarm state: on

\_delay\_ms(200);

Alarm=0; //set pumping moment to 0

break;

}

if (PINC==(PINC&~(1<<PC2))) //if PC2 pressed

{

alarmode=0; //Alarm state: off

\_delay\_ms(200);

goto Here; //Return to main menu

}

if (PINC==(PINC&~(1<<PC4))) //if PC4 pressed

{

\_delay\_ms(200);

Alarm=0; //set pumping moment to 0

goto Here; //Return to main menu

}

}

LCDClear();

LCDGotoXY(0,0);

LCDWriteString("Time:ss/mm/hh"); //Display timer-pumping setting menu

LCDGotoXY(2,1);

LCDWriteString(":");

LCDGotoXY(5,1);

LCDWriteString(":"); //print " : :"

uint8\_t cout4=0;

while (1)

{

if (cout4==0) //get first second digit

{

LCDGotoXY(0,1);

if (checkkey()==1)

{

key = KEYPAD\_GetKey();

itoa(key-48,Showkey,10); //convert key from integer to ascii for displaying

LCDWriteString(Showkey);

\_delay\_ms(100);

cout4++;

}

}

if (cout4==1) //get second second digit

{

LCDGotoXY(1,1);

if (checkkey()==1)

{

key = KEYPAD\_GetKey();

itoa(key-48,Showkey,10); //convert key from integer to ascii for displaying

LCDWriteString(Showkey);

\_delay\_ms(100);

cout4++;

}

}

if (cout4==2) //get first minute digit

{

LCDGotoXY(3,1);

if (checkkey()==1)

{

key = KEYPAD\_GetKey();

itoa(key-48,Showkey,10); //convert key from integer to ascii for displaying

Alarm=Alarm+(key-48)\*10; //set the first digit of minute alarm

LCDWriteString(Showkey);

\_delay\_ms(100);

cout4++; //go to next digit position

}

}

if (cout4==3) //get second minute digit

{

LCDGotoXY(4,1);

if (checkkey()==1)

{

key = KEYPAD\_GetKey();

itoa(key-48,Showkey,10); //convert key from integer to ascii for displaying

Alarm=Alarm+(key-48); //set the second digit of minute alarm

LCDWriteString(Showkey);

\_delay\_ms(100);

cout4++;

}

}

if (cout4==4) //get first hour digit

{

LCDGotoXY(6,1);

if (checkkey()==1)

{

key = KEYPAD\_GetKey();

itoa(key-48,Showkey,10); //convert key from integer to ascii for displaying

Alarm=Alarm+(key-48)\*60\*10; //set the first digit of hour alarm

LCDWriteString(Showkey);

\_delay\_ms(100);

cout4++;

}

}

if (cout4==5) //get second hour digit

{

LCDGotoXY(7,1);

if (checkkey()==1)

{

key = KEYPAD\_GetKey();

itoa(key-48,Showkey,10); //convert key from integer to ascii for displaying

Alarm=Alarm+(key-48)\*60; //set the second digit of hour alarm

LCDWriteString(Showkey);

\_delay\_ms(100);

cout4++;

}

}

if (cout4==6) //if already get enough 6 digits

{

\_delay\_ms(3000); //delay 3 seconds

LCDtt=1; //go back to main menu

break;

}

}

Here:LCDtt=1; //go back to main menu

}

int checkkey() //

{

key= KEYPAD\_GetKey();

if ((key-48>=0) && (key-48<10)) //check number - ASCII table

{

return 1;

}

else {return 0;}

}

int checknumber(int ndata) //correct RTC result

{

if ((ndata>=16)&&(ndata<=25))

{

return 6;

}

if ((ndata>=32)&&(ndata<=41))

{

return 12;

}

if ((ndata>=48)&&(ndata<=57))

{

return 18;

}

if ((ndata>=64)&&(ndata<=73))

{

return 24;

}

if ((ndata>=80)&&(ndata<=89))

{

return 30;

}

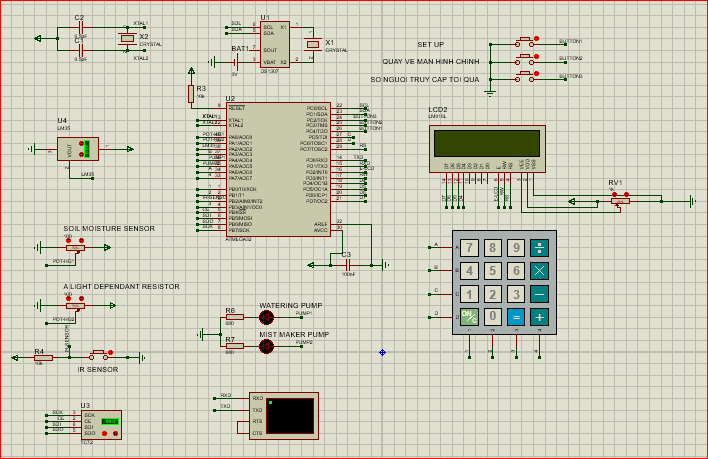
if ((ndata>=0)&&(ndata<=9))

{

return 0;

}

}

**Proteus Circuit:**

**Start**

**Configure direction of Atmega32 pins (inputs – outputs)**

**Initialize LCD, ADC, SPI, TC72, RTC, Keypad, Timer Interrupt, UART**

**While**

**(True)**

**Get time (RTC)**

**Check daytime/nighttime (LDR)**

**Disable INT2,**

**Deactivate IR sensor (not count people)**

**Reset people counter**

**Enable INT2 (Turn on LED for each visitor),**

**Activate IR sensor (count people)**

**Increment people counter**

**Day**

**Night**

**Activate**

**Water pump**

**Check timer-pumping mode (using input from Keypad)**

**Time up**

**Time not up**

**Monitor Temperature and Moisture**

**(compare with thresholds input from UART terminal)**

**Activate Pumping**

**(Water or mist maker or both)**

**Lower than threshold**

**>= threshold**

**If people-monitoring button pressed**

**LCD display people monitor screen**

**If main-screen button pressed**

**LCD Display main screen**

**If status - setup button pressed**

**LCD display setting up temperature & moisture screen**

**Get data from UART connect**

**If status - setup button pressed**

**LCD display timer-pump setup screen**

**Get data from keypad**

**True**

**True**

**True**

**True**

**False**

**False**

**False**

**False**

**Flowchart:**