Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 2, No. 1, February : 2023

AUTONOMOUS AGRICULTURAL ECOSYSTEM MONITORING

Dr. P.V. Kishore Professor, St. Peters Engineering College, Maisammaguda, Hyderabad.
Mr. M. Krishna Assistant Professor, St. Peters Engineering College, Maisammaguda, Hyderabad.
Mrs. K. Laxmi Krishna Assistant Professor, St. Peters Engineering College, Maisammaguda, Hyderabad.

ABSTRACT

More than two-thirds of freshwater consumed worldwide are used for irrigation, and large quantities of freshwater can be saved by improving the efficiency of irrigation systems. Irrigation control systems deployed in agriculture can substantially be enhanced by implementing intelligent monitoring techniques enabling automated sensing and continuous analyses of actual soil parameter. Automatically scheduling irrigation events based on soil moisture measurements has been proven an effective means to reduce freshwater consumptionand irrigation costs, while maximizing the crop yield. Focusing on decentralized autonomoussoil moisture monitoring, this project presents the design, the implementation, and the validationof a low-cost remote monitoring system for agricultural ecosystems. The prototype monitoring system consists of a number of intelligent wireless sensor nodes that are distributed in the observed environment. The sensor nodes are connected to an Internet-enabled computer system, which is installed on site for disseminating relevant soil information and providing remote access to the monitoring system. Autonomous software programs, labeled “mobile software agents”, are embedded into the wireless sensor nodes to continuously analyze the soil parameters and to autonomously trigger irrigation events based on the actual soil conditions and on weather data integrated from external sources

INTRODUCTION

1.1 Introduction:

The main aim of this paper is to design low cost Agriculture ecosystem for irrigation to reduce the manual monitoring of the field using sensors. The data is also displayedon LCD.

This paper makes a use of PIC Microcontroller the main controlling device of the project. Soil moisture sensor is uses to detect the moisture content in soil. Water level senor is uses to detect the water level in tank. LM5 Temperature sensor is uses to detect the temperature of motor if the motor works

without water motor will get heat. The status of the project will display on LCD.

The controlling device of the whole system is PIC microcontroller. Whenever the sensors unit gets the input from respected sensors like temperature sensor, Water level indicator and soil moisture sensor, these inputs are fed to the microcontroller. The microcontroller performs appropriate task related to the data received like motor ON/OFF control. If soil moisture is low or water level is low this system will switch on the water motor. If water level is at appropriate level then motor OFF automatically. If temperature of the motor crosses the set limit then microcontroller will switch OFF the motor automatically. The appropriate values displayed on LCD.

EMBEDDED SYSTEMS

2.1 Embedded Systems:

An embedded system is a computer system designed to perform one or a few dedicated functions often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. By contrast, a general-purpose computer, such as a personal computer (PC), is designed to be flexible and to meet a wide range of end-user needs. Embedded systems control many devices in common use today.

Embedded systems are controlled by one or more main processing cores that are typically either microcontrollers or digital signal processors (DSP). The key characteristic, however, is being dedicated to handle a particular task, which may require very powerful processors. For example, air traffic control systems may usefully be viewed as embedded, even though they involve mainframe computers and dedicated regional and national networks between airports and radar sites. (Each radar probably includes one or more embedded systemsof its own.)

Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Physically embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, or the systems controlling nuclear power plants. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted insidea large chassis or enclosure.

In general, "embedded system" is not a strictly definable term, as most systems have some element of extensibility or programmability. For example, handheld computers share some elements with embedded systems such as the operating systems and microprocessors which power them, but they allow different applications to be loaded and peripherals to be connected. Moreover, even systems which don't expose programmability as a primary feature generally need to support software updates. On a continuum from "general purpose" to "embedded", large application systems will have subcomponents at most points even if the system as a whole is "designed to perform one or a few dedicated functions", and is thus appropriate to call "embedded". A modern example of embedded system is shown in fig: 2.1.

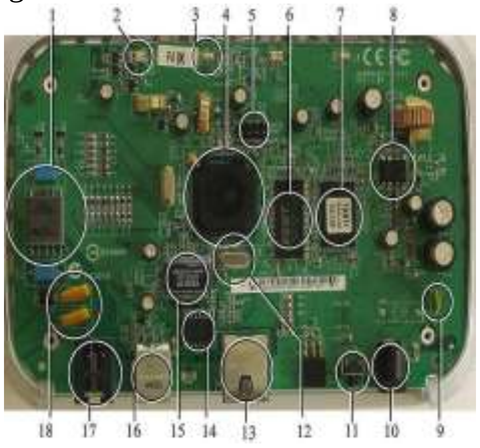


Fig 2.1:A modern example of embedded system

Labeled parts include microprocessor (4), RAM (6), flash memory (7).Embedded systems programming is not like normal PC programming. In many ways, programming for an embedded system is like programming PC 15 years ago. The hardware for the system is usually chosen to make the device as cheap as possible. Spending an extra dollar a unit in order to make things easier to program can cost millions. Hiring a programmer for an extra month is cheap in comparison. This means the programmer must make do with slow processors and low memory, while at the same time battling a need for efficiency not seen in most PC applications. Below isa list of issues specific to the embedded field.

2.2 Software Architecture:

There are several different types of software architecture in common use.

Simple Control Loop:In this design, the software simply has a loop. The loop calls subroutines, each ofwhich manages a part of the hardware or software.

Interrupt Controlled System:Some embedded systems are predominantly interrupt controlled. This means that tasks performed by the system are triggered by different kinds of events. An interrupt could begenerated for example by a timer in a predefined frequency, or by a serial port controller receiving a byte. These kinds of systems are used if event handlers need low latency and the event handlers are short and simple.

Usually these kinds of systems run a simple task in a main loop also, but this task is not very sensitive to unexpected delays. Sometimes the interrupt handler will add longer tasks to a queue structure.

Later, after the interrupt handler has finished, these tasks are executed by the main loop. This method brings the system close to a multitasking kernel with discrete processes.

Cooperative Multitasking:A non-preemptive multitasking system is very similar to the simple control loop scheme, except that the loop is hidden in an API. The programmer defines a series of tasks, and each task gets its own environment to "run" in. When a task is idle, it calls an idle routine, usually called "pause", "wait", "yield", "nop" (stands for no operation), etc.The advantages and disadvantages are very similar to the control loop, except that adding new software is easier, by simply writing a new task, or adding to the queue-interpreter.

Primitive Multitasking: In this type of system, a low-level piece of code switches between tasks or threads based on a timer (connected to an interrupt). This is the level at which the system is generally considered to have an "operating system" kernel. Depending on how much functionality is required, it introduces more or less of the complexities of managing multiple tasks running conceptually in parallel.

As any code can potentially damage the data of another task (except in larger systems using an MMU) programs must be carefully designed and tested, and access to shared data must be controlled by some synchronization strategy, such as message queues, semaphores or a non- blocking synchronization scheme.

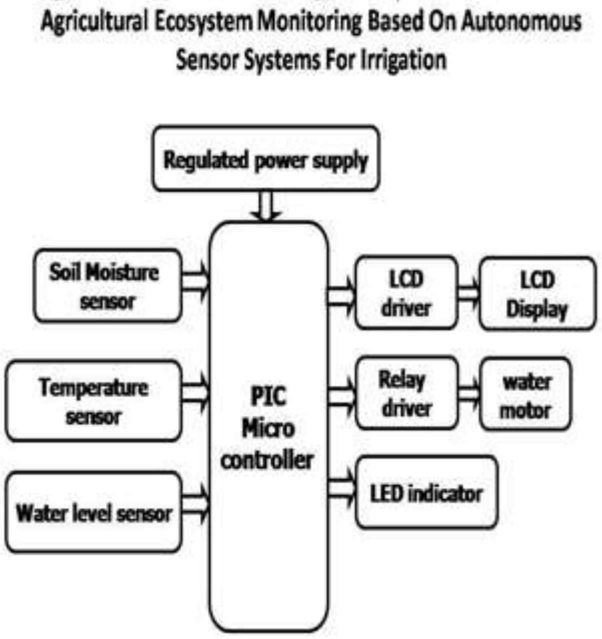
Because of these complexities, it is common for organizations to buy a real-time operating system, allowing the application programmers to concentrate on device functionality rather than operating system services, at least for large systems; smaller systems often cannot afford the overhead associated with a generic real time system, due to limitations regarding memory size, performance, and/or battery life.

Microkernels And Exokernels:

A microkernel is a logical step up from a real-time OS. The usual arrangement is that the operating system kernel allocates memory and switches the CPU to different threads ofexecution. User mode processes implement major functions such as file systems, network interfaces, etc.

In general, microkernels succeed when the task switching and intertask communicationis fast, and fail when they are slow. Exokernels communicate efficiently by normal subroutine calls. The hardware and all the software in the system are available to, and extensible by application programmers. Based on performance, functionality, requirement the embedded systems are divided into three categories:

3.0 HARDWARE DESCRIPTION



In this chapter the block diagram of the project and design aspect of independentmodules are considered. Block diagram is shown in fig: 3.1:

The main blocks of this paper

1. Regulated Power Supply.
2. PIC Microcontroller.
3. Soil moisture sensor.
4. Temperature sensor.
5. Water level sensor.
6. Water motor.
7. LCD display.
8. Reset button.
9. Crystal oscillator.
10. LED indicators.

4.0 SOFTWARE DESCRIPTION

This work is implemented using following software's:Express PCB – for designing circuit
PIC C compiler - for compilation part

4.1 Express PCB:

Breadboards are great for prototyping equipment as it allows great flexibility to modifya design when needed; however, the final product of a project, ideally should have a neat PCB, few cables, and survive a shake test. Not only is a proper PCB neater but it is also more durableas there are no cables which can yank loose.

Express PCB is a software tool to design PCBs specifically for manufacture by the company Express PCB (no other PCB maker accepts Express PCB files). It is very easy touse, but it does have several limitations.

It can be likened to more of a toy then a professional CAD program.It has a poor part library (which we can work around)
It cannot import or export files in different formats

It cannot be used to make prepare boards for DIY production

Express PCB has been used to design many PCBs (some layered and with surface-mount parts. Print out PCB patterns and use the toner transfer method with an Etch Resistant Pen to make boards. However, Express PCB does not have a nice print layout. Here is the procedure to design in Express PCB and clean up the patterns so they print nicely.

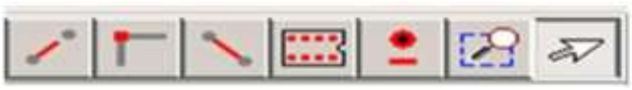
Preparing Express PCB for First Use:

Express PCB comes with a less then exciting list of parts. So, before any projectis started head over to Audio logical and grab the additional parts by morsel, ppl, and tangent,

and extract them into your Express PCB directory. At this point start the program and get readyto setup the workspace to suit your style.In this menu, setup the units for "mm" or "in" depending on how you think, and click "see through the top copper layer" at the bottom. The standard color scheme of red and green is generally used but it is not as pleasing as red and blue.

4.1.2 The Interface:

When a project is first started you will be greeted with a yellow outline. This yellow outline is the dimension of the PCB. Typically, after positioning of parts and traces, move them to their final position and then crop the PCB to the correct size. However, in designing a board with a



certain size constraint, crop the PCB to the correct size before starting.Fig: 4.1 show the toolbar in which each button has the following functions:

Fig 4.1: Tool bar necessary for the interface

The select tool: It is fairly obvious what this does. It allows you to move and manipulate parts. When this tool is selected the top toolbar will show buttons to move traces to the top / bottom copper layer, and rotate buttons.The zoom to selection tool: does just that.

The place pad: button allows you to place small soldier pads which are useful for board connections or if a part is not in the part library but the part dimensions are available. When this tool is selected the top toolbar will give you a large selection of round holes, square holes and surface mount pads.The place component: tool allows you to select a component from the top toolbar and then by clicking in the workspace places that component in the orientation chosen using the buttons next to the component list. The components can always be rotated afterwards with the select tool ifthe orientation is wrong.The place trace: tool allows you to place a solid trace on the board of varying thicknesses. The top toolbar allows you to select the top or bottom layer to place the trace on.The Insert Corner in trace: button does exactly what it says. When this tool is selected, clickingon a trace will insert a corner which can be moved to route around components and other traces.The remove a trace button is not very important since the delete key will achieve the sameresult.

4.1.3 Design Considerations:

Before starting a project there are several ways to design a PCB and one must be chosen to suit the project’s needs. Single sided, or double sided?

When making a PCB you have the option of making a single sided board, or a double- sided board. Single sided boards are cheaper to produce and easier to etch, but much harder to design for large projects. If a lot of parts are being used in a small space it may be difficult to make a single sided board without umpiring over traces with a cable. While there’s technically nothing wrong with this, it should be avoided if the signal travelling over the traces is sensitive (e.g., audio signals).

A double-sided board is more expensive to produce professionally, more difficult to etch on a DIY board, but makes the layout of components a lot smaller and easier. It should be noted that if a trace is running on the top layer, check with the components to make sure you can get to its pins with a soldering iron. Large capacitors, relays, and similar parts which don’t have axial leads can NOT have traces on top unless boards are plated professionally. Ground-plane or other special purposes for one side When using a double-sided board, you must consider which traces should be on whatside of the board. Generally, put power traces on the top of the board, jumping only to the bottom if a part cannot be soldiered onto the top plane (like a relay), and vice- versa. Some projects like power supplies or amps can benefit from having a solid plane to use for ground. In power supplies this can reduce noise, and in amps it minimizes the distance between parts and their ground connections, and keeps the ground signal as simple as possible. However, care must be taken with stubborn chips such as the TPA6120 amplifier from TI. The TPA6120 datasheet specifies not to run a ground plane under the pins or signal traces of thischip as the capacitance generated could affect performance negatively.

4.2 PIC Compiler:

PIC compiler is software used where the machine language code is written and compiled. After compilation, the machine source code is converted into hex code which is to be dumped into the microcontroller for further processing. PIC compiler also supports C language code. It’s important that you know C language for microcontroller which is commonly known as Embedded C. As we are going to use PIC Compiler, hence we also call it PIC C. The PCB, PCM, and PCH are separate compilers. PCB is for 12-bit opcodes, PCM is for 14- bitopcodes, and PCH is for 16-bit opcode PIC microcontrollers. Due to many similarities, all three compilers are covered in this reference manual. Features and limitations that apply to only specific microcontrollers are indicated within. These compilers are specifically designed to meet the unique needs of the PIC microcontroller. This allows developers to quickly design applications software in a more readable, high-level language. When compared to a more traditional C compiler, PCB, PCM, and PCH have some limitations. As an example of the limitations, function recursion is not allowed.

This is due to the fact that the PIC has no stack to push variables onto, and also because of the way the compilers optimize the code. The compilers can efficiently implement normal C constructs, input/output operations, and bit twiddling operations. All normal C data types are supported along with pointers to constant arrays, fixed point decimal, and arrays of bits.

PIC C is not much different from a normal C program. If you know assembly, writing a C program is not a crisis. In PIC, we will have a main function, in which all your application specific work will be defined. In case of embedded C, you do not have any operating system running in there. So, you have to make sure that your program or main file should never exit. This can be done with the help of simple while (1) or for (; ;) loop as they are going to run infinitely. We have to add header file for controller you are using, otherwise you will not be able to access registers related to peripherals.

4.3 Procedural steps for compilation, simulation and dumping:

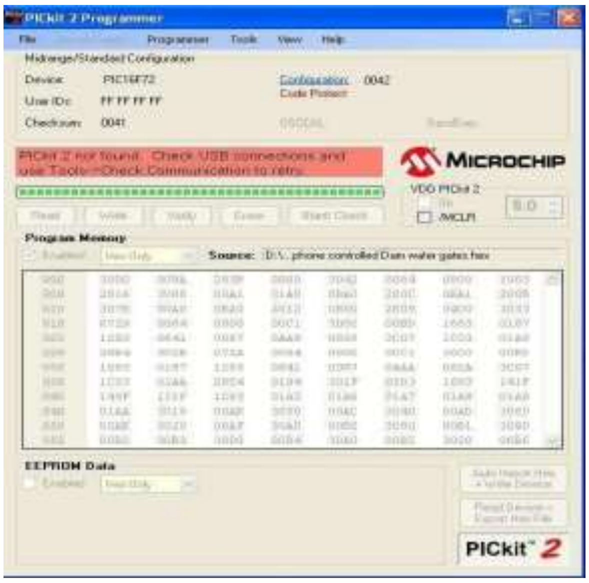
4.3.1 Compilation and simulation steps:

For PIC microcontroller, PIC C compiler is used for compilation. The compilation steps are as follows:

Open PIC C compiler.You will be prompted to choose a name for the new project, so create a separate folder where all the files of your project will be stored, choose a name and click save.Click Project**,** New, and something the box named 'Text1' is where your code should be written later.Now you have to click 'File, save as' and choose a file name for your source code ending withthe letter '.c'. You can name as 'projects' for example and click save. Then you have to add this file to your project work.You can then start to write the source code in the window titled 'project's' then before testingyour source code; you have to compile your source code, and correct eventual syntax errors.By clicking on compile option .hex file is generated automatically.This is how we compile a program for checking errors and hence the compiled program is savedin the file where we initiated the program.After compilation, next step is simulation. Here first circuit is designed in Express PCB using Proteus 7 software and then simulation takes place followed by dumping. The simulation steps are as follows:Open Proteus 7 and click on IS1S6.Now it displays PCB where circuit is designed using microcontroller. To design circuit components are required. So, click on component option.Now click on letter 'p', then under that select PIC16F877A, other components related to the project and click OK. The PIC 16F877A will be called your "'Target device", which is the final destination of your source code.

4.4.2 Dumping steps:

The steps involved in dumping the program edited in proteus 7 to microcontroller are shown below: Initially before connecting the program dumper to the microcontroller kit the window isappeared as shown below.



Midrange/Standard Configuration
Device: PIC16F72
User IDs: FF FF FF FF
Checksum: 0041
Configuration: 0042
Code Protect

PICkit 2 not found. Check USB connections and use Tools->Check Communication to retry.

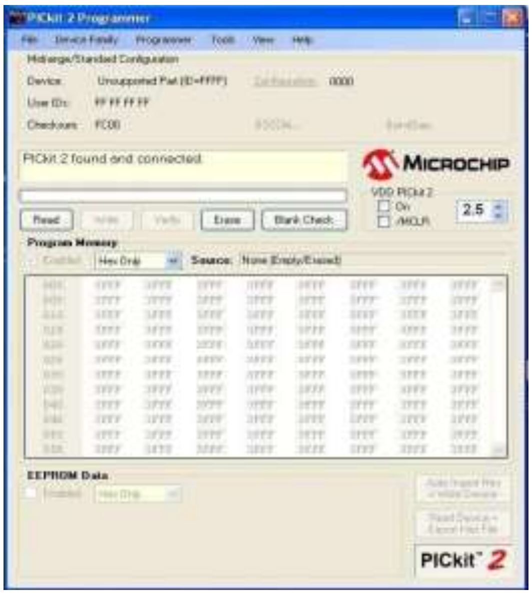
VDD PICkit 2
/MCLR
5.0

Program Memory
Source: D:\...phone controlled Dam water gates.hex

EEPROM Data

Fig 4.2: Picture of program dumper window

1. Select Tools option and click on Check Communication for establishing a connection as shownin below window
2. connecting the dumper properly to the microcontroller kit the window is appeared asshown below.
3. by selecting the Tools option and clicking on Check Communication the microcontroller gets recognized by the dumper and hence the window is as shown below.



Midrange/Standard Configuration
Device: Unsupported Part (ID=FFFF)
User IDs: FF FF FF FF
Checksum: FC00
Configuration: 0000

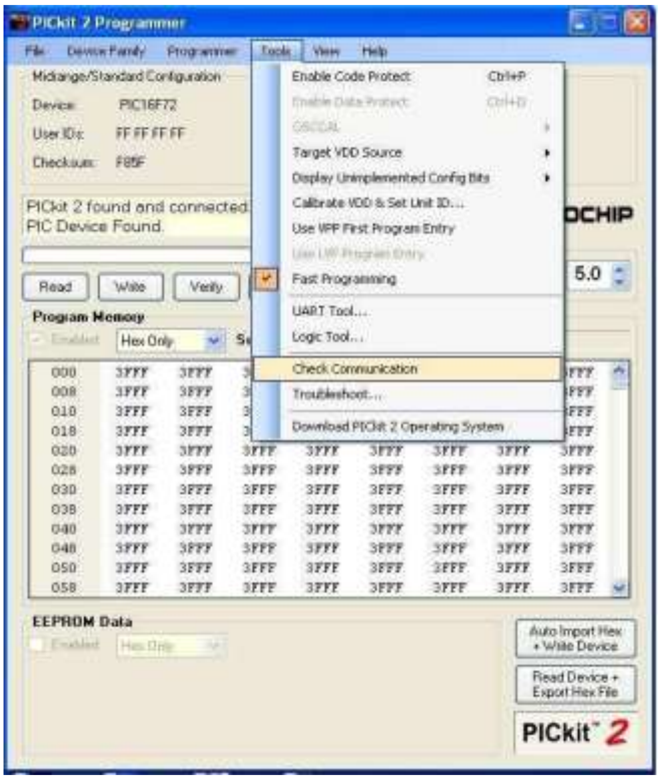
PICkit 2 found and connected.

VDD PICkit 2
On
/MCLR
2.5

Program Memory
Hex Only
Source: None (Empty/Erased)

EEPROM Data

Fig 4.3: Picture of checking communications before dumping program intomicrocontroller


PICkit 2 Programmer
File
Device Family
Programmer
Tools
View
Help
Enable Code Protect Ctrl+P
Enable Data Protect Ctrl+D
OSCCAL
Target VDD Source
Display Unimplemented Config Bits
Calibrate VDD & Set Unit ID...
Use VPP First Program Entry
Use LVP Program Entry
Fast Programming
UART Tool...
Logic Tool...
Check Communication
Troubleshoot...
Download PICkit 2 Operating System
OCHIP
5.0
Read
Write
Verify
Hex Only
Enabled
Auto Import Hex + Write Device
Read Device + Export Hex File
PICkit 2

Midrange/Standard Configuration

Device: PIC16F72
User IDs: FF FF FF FF
Checksum: F85F

PICkit 2 found and connected.
PIC Device Found.

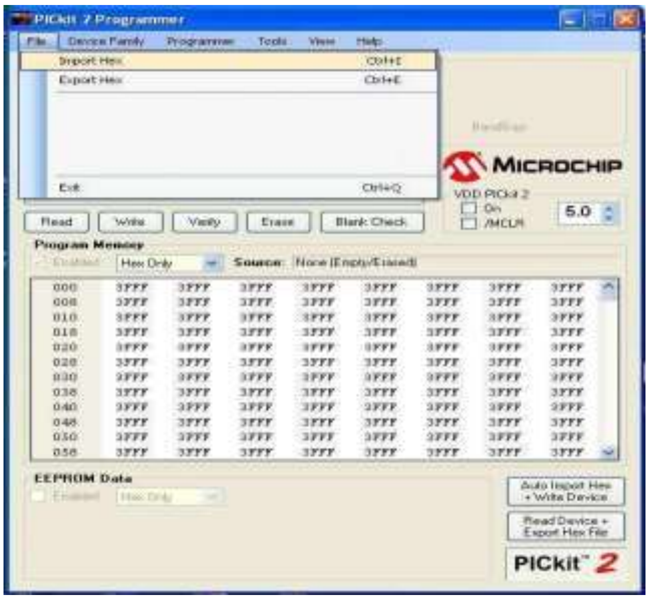
Program Memory

000	3FFF	3FFF	3					FFF
008	3FFF	3FFF	3					FFF
010	3FFF	3FFF	3					FFF
018	3FFF	3FFF	3					FFF
020	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
028	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
030	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
038	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
040	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
048	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
050	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
058	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF

EEPROM Data

Fig 4.4: Picture of dumper recognition to microcontroller

4. Import the program which is '. hex' file from the saved location by selecting File option and clicking on 'Import Hex' as shown in below window.



Program Memory

Source: None (Empty/Erased)

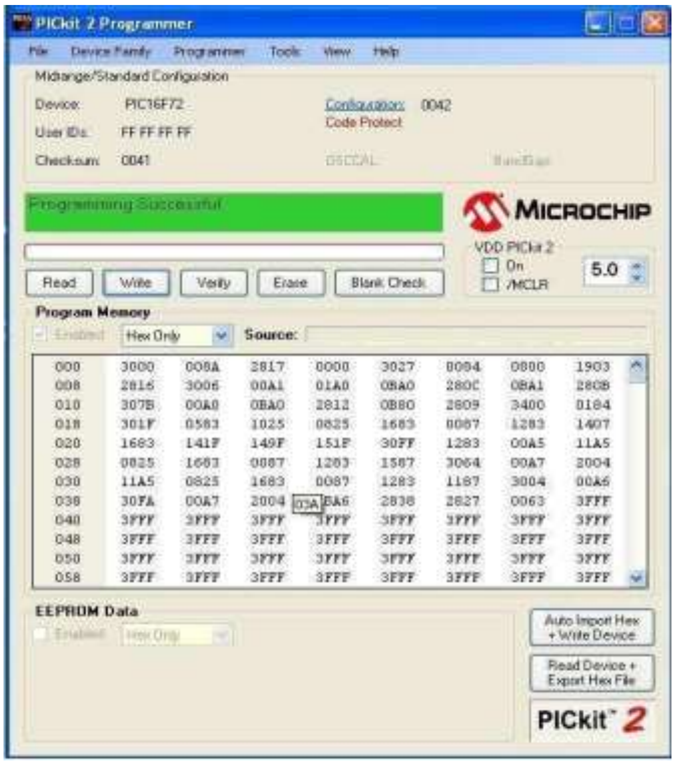
000	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
008	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
010	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
018	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
020	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
028	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
030	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
038	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
040	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
048	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
050	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF
058	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF	3FFF

EEPROM Data

Fig 4.5: Picture of program importing into the microcontroller

5. After clicking on 'Import Hex' option we need to browse the location of our program and click the 'prog.hex' and click on 'open' for dumping the program into the microcontroller.

6. After the successful dumping of program the window is as shown below.


PICkit 2 Programmer
File Device Family Programmer Tools View Help
MICROCHIP
Read Write Verify Erase Blank Check
Auto Import Hex + Write Device
Read Device + Export Hex File
PICkit 2

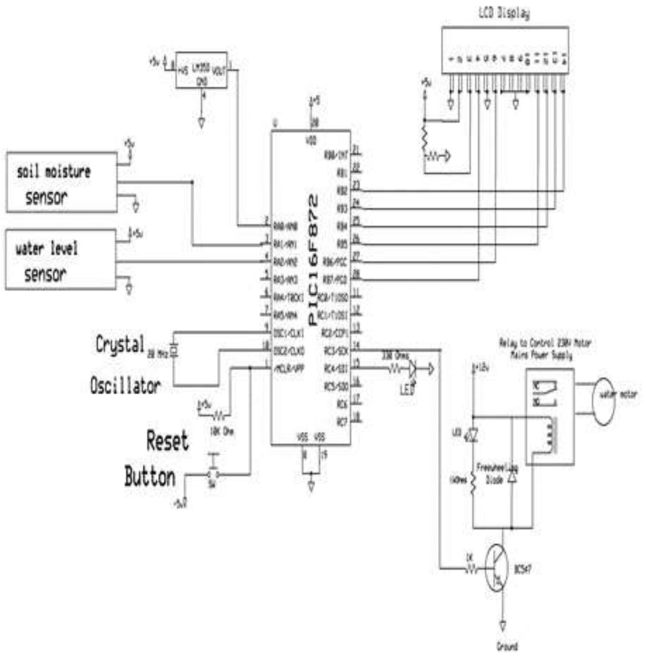
Midrange/Standard Configuration
Device: PIC16F72
User IDs: FF FF FF FF
Checksum: 0041
Configuration: 0042
Code Protect
Programming Successful
VDD PICkit 2 ☐ On ☐ /MCLR 5.0
Program Memory
Hex Only Source:

```
000   3000  008A  2817  0000  3027  0084  0800  1903
008   2816  3006  00A1  01A0  0BA0  280C  0BA1  280B
010   307B  00A0  0BA0  2812  0B80  2809  3400  0184
018   301F  0583  1025  0825  1683  0087  1283  1407
020   1683  141F  149F  151F  30FF  1283  00A5  11A5
028   0825  1683  0087  1283  1587  3064  00A7  2004
030   11A5  0825  1683  0087  1283  1187  3004  00A6
038   30FA  00A7  2004 03A BA6  2838  2827  0063  3FFF
040   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
048   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
050   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
058   3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF  3FFF
```

EEPROM Data
Hex Only

Fig 4.6: Picture after program dumped into the microcontroller

schematic diagram and interfacing of PIC microcontroller with each moduleis considered**.**



The above schematic diagram of **Agricultural ecosystem monitoring based onautonomous sensor systems for irrigation** explains the interfacing section of each component with PIC microcontroller.

5.0 RESULTS

- The controlling device of the whole system is PIC microcontroller. Whenever the sensors unit gets the input from respected sensors like temperature sensor, Water level indicator and soil moisture sensor, these inputs are fed to the microcontroller. The microcontroller performs appropriate task related to the data received like motor ON/OFF control. If soil moisture is low or water level is low this system will switch on the water motor. If water level is at appropriate level, then motor OFF automatically. If temperature of the motor crosses the set limit, then microcontroller will switch OFF the motor automatically. The appropriate values displayed on LCD.To perform this task, Microcontroller is programmed using embedded C language and pic c compilers.

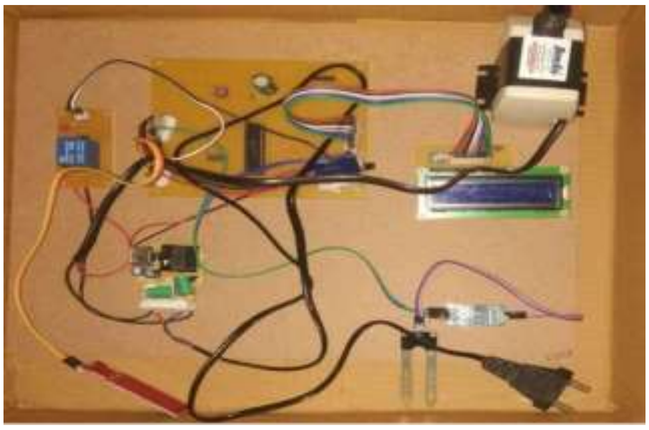


Fig 7.1 ideal condition

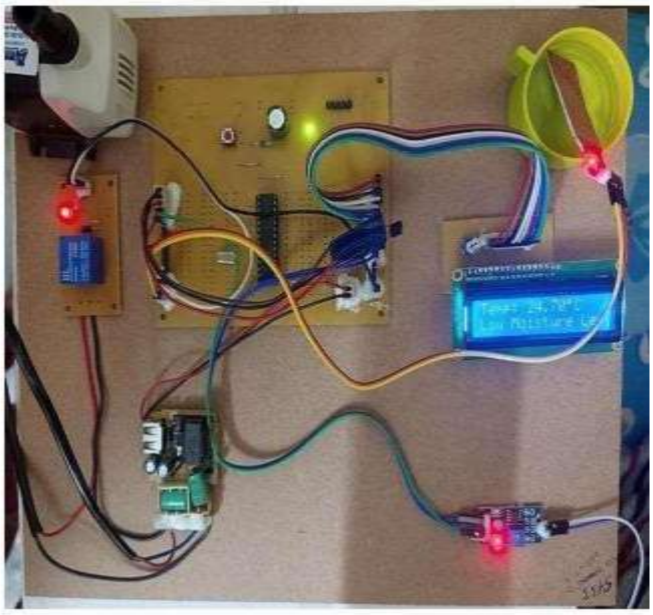


Fig 7.2 Working Condition

6.0 CONCLUSION AND FUTURE SCOPE

Conclusion:

Integrating features of all the hardware components used have been developed in it. Presence of every module has been reasoned out and placed carefully, thus contributing to the best working of the unit. Secondly, using highly advanced ICs with the help of growing technology, the project has been successfully implemented. Thus, the project has been successfully designed and tested.

Future Scope:

The work can be extended by adding GSM to get alert messages.We can add Raspberry pi processor and some sensors like NPK sensor, PH sensor, So the usercan check his field every time and keep the field safe.The work can be extended by adding **SOLAR panel it is free source from sun**.

REFERENCES

[1] "Global Environment Outlook: Environment for Development (GEO- 4)", United Nations Environment Programmed (UNEP), Nairobi, Kenya, 2007.
[2] "Water Facts and Trends", World Business Council for Sustainable Development (WBCSD), Geneva, Switzerland, 2006.
[3] S. Maxwell, "Historical Water Price Trends", AWWA Journal, vol. 102, no. 4, pp. 24-28, 2010.
[4] The United Nations World Water Development Report 3", The United Nations Educational, Scientific and Cultural Organization (UNESCO), Paris, France, 2009.
[5] F.S. Zarzuela, A.G. Sastra and GA. Clark, "Irrigation System Controllers". University of Florida, Institute of Food and Agricultural Sciences, FL, USA, 2008.

[6] A.G. Sastra, B.J. Bowman, G.A. Clark, D.Z. Haman, D.S. Harrison, F.T. Izumo, DJ. Pitts and F.S. Zarzuela, "Efficiencies of Florida Agricultural Irrigation Systems", University of Florida, Institute of Food and Agricultural Sciences, FL, USA, 2008.

[7] MD. Dukes, "Smart Irrigation Controllers: What Makes an Irrigation Controller Smart?", University of Florida, Institute of Food and Agricultural Sciences, FL, USA, 2009.

[8] MD Dukes, M. Shedd and B. Cardenas-LaPlace, "Smart Irrigation Controllers: How De Soil Moisture Sensor (SMS) Irrigation Controllers Work?", University of Florida, Institute of Food and Agricultural

[9] "Weather and Soil Moisture Based Landscape Irrigation Scheduling Devices", Technical Review Report - 2nd Edition, U.S. Department of the Interior, Bureau of Reclamation, Washington, DC, USA, 2007.

[10] N. Wang, N. Zhang and M. Wang, "Recent Wireless sensors in agriculture and food industry Recent development and future perspective", Computers and Electronics in Agriculture, vol. 50, no. 1, pp. 1-14, 2006. -

[11] KH. Law, K. Smartly and Y. Wang, "Sensor Data Management Technologies". In: Wang, ML, Lynch, J.P. and Sohn, H. (eds.), "Sensor Technologies for Civil Infrastructures: Performance Assessment and Health Monitoring", Saws ton, UK: Woodhead Publishing, Ltd. (in press).

[12] K. Smartly and K.H. Law, "Coupling Wireless Sensor Networks and Autonomous Software for Integrated Soil Moisture Monitoring", The 10th International Conference Hydro informatics. Hamburg, on Germany, July 14, 2012

[13] K. Smartly, K. Georgieva, M. König and KH Law, "Monitoring of Slope Movements coupling Autonomous Wireless Sensor Networks and Web Services", The First International Conference on Performance- Based Life-Cycle Structural Engineering Hong Kong, China, December 5, 2012

[14] Smartly, K., Law, KH and König, M., Autonomous Structural Condition Monitoring based on Dynamic Code Migration and Cooperative Information Processing in Wireless Sensor Networks". The 8th International Workshop on Structural Health Monitoring 2011. Stanford, CA, USA, September 13, 2011.