Automation in Agriculture Using IoT for Reduction in Human Error

Dharshni Prabha P
Department of Electrical and Electronics Engineering
Paavai Engineering College
Namakkal, Tamilndau. India.

Dr.Jagatheesan K, Associate Professor
Department of Electrical and Electronics Engineering,
Paavai Engineering College,
Namakkal, Tamilndau. India.

ABSTRACT

Minimizing human labor Automation can help the minimization of human resource and human error. Quick Access Crop and soil health can be remotely monitored through any device and at any region. Time Efficient Automatic report generation and remote monitoring can save time and effort from farmers. Efficient Communication Using Android and iOS app platform a community of farmers, students and enthusiasts can share their work and new methodologies for agricultural growth. Analytics Wide range of analysis can be done including avg. rainfall, soil health gradient. The applications of IoT-based smart farming not only target conventional, large farming operations, but could also be new levers to uplift other growing or common trends in agricultural like organic farming, family farming (complex or small spaces, particular cattle and/or cultures, preservation of particular or high-quality varieties, etc.), and enhance highly transparent farming.

Keywords—IoS, location, optimization, IoT

I. INTRODUCTION*:*

In the world of Innovative and automotive world everything is getting computerized. Smart farming is a capital-intensive and hi-tech system of growing food cleanly and sustainable for the masses. It is the application of modern ICT (Information and Communication Technologies) into agriculture.

In terms of environmental issues, IoT-based smart farming can provide great benefits including more efficient water usage, or optimization of inputs and treatments. Now, let's discuss the major applications of IoT-based smart farming that are revolutionizing agriculture.

II. LITERATURE REVIEW

Internet-of-Things (IoT) based Smart Agriculture: Towards Making the Fields Talk

Despite the perception people may have regarding the agricultural process, the reality is that today's agriculture industry is data-cantered, precise, and smarter than ever. The rapid emergence of the Internet-of-Things (IoT) based technologies redesigned almost every industry including "smart agriculture" which moved the industry from statistical to quantitative approaches. Such revolutionary changes are shaking the existing agriculture methods and creating new opportunities along a range of challenges. This article highlights the potential of wireless sensors and IoT in agriculture, as well as the challenges expected to be faced when integrating this technology with the traditional farming practices. IoT devices and communication techniques associated with wireless sensors encountered in agriculture applications are analyzed in detail. What sensors are available for specific agriculture application, like soil preparation, crop status, irrigation, insect and pest detection are listed. How this technology helping the growers throughout the crop stages, from sowing until harvesting, packing and transportation is explained. Furthermore, the use of unmanned aerial vehicles for crop surveillance and other favorable applications such as optimizing crop yield is considered in this article. State-of-the-art IoT-based architectures and platforms used in

agriculture are also highlighted wherever suitable. Finally, based on this thorough review, we identify current and future trends of IoT in agriculture and highlight potential research challenges.

An Internet of things (IoT) architecture for Smart Agriculture

Agriculture is one major and important sector for the growth of economy for any country. As per the current scenarios, various problems are present in agriculture like techniques which are used currently are not efficient, requirement of larger manpower and appropriate time for irrigation and spreading of fertilizer to yield. Internet of Things (IoT) is latest technology for smart farming to enhance efficiency, productivity and resolve various issues present in agriculture. IoT network consist of various sensor node which is used to monitor soil acidity level, temperature, and other variables. In this paper, the steps involved for agriculture are discussed and mainly focus on use of IoT in agriculture i.e. in proposed architecture which leads to growth of agriculture exponentially and the economy.3. Sumi and Ranga suggested a smart traffic management approach for nations based on the IoT and the vehicular ad hoc network principles (VANET).

Mobile based Horne Automation using Internet of Things (IoT)

Availability 01' high speed mobile networks like 3G and Long Term Evolution(LTE) coupled with cheaper and accessible smart phones, mobile industry has seen a tremendous growth in terms 01' providing various services and applications at the finger tips 01' the citizens. Internet 01' Things(loT) is one 01' the promising technologies which can be used for connecting, controlling and managing intelligent objects which are connected to Internet through an IP address. Applications ranging from smart governance, smart education, smart agriculture, smart health care, smart hornes etc can use loT tor effective delivery 01' services with out manual intervention in a more effective manner. This paper discusses about loT and how it can be used for realizing smart Horne automation using a micro-controller based Arduino board and Android mobile app. In this paper, two prototypes namely horne automation using Bluetooth in an indoor environment and horne automation using Ethernet in an outdoor environment are presented.

IOT Based Smart Agriculture System

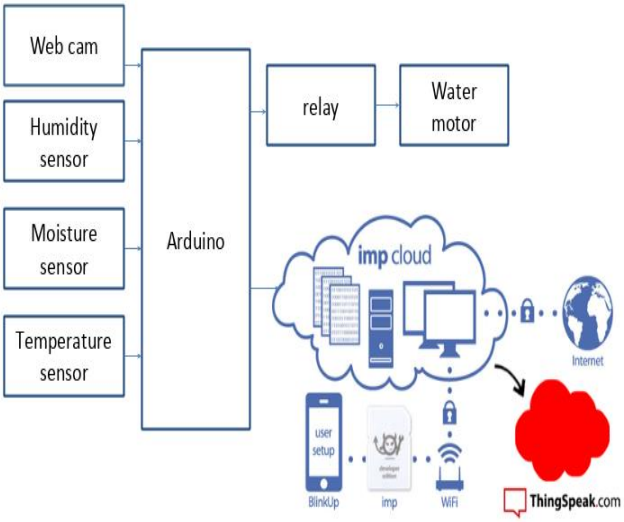
Smart agriculture is an emerging concept, because IOT sensors are capable of providing information about agriculture fields and then act upon based on the user input. In this Paper, it is proposed to develop a Smart agriculture System that uses advantages of cutting edge technologies such as Arduino, IOT and Wireless Sensor Network. The paper aims at making use of evolving technology i.e. IOT and smart agriculture using automation. Monitoring environmental conditions is the major factor to improve yield of the efficient crops.

The feature of this paper includes development of a system which can monitor temperature, humidity, moisture and even the movement of animals which may destroy the crops in agricultural field through sensors using Arduino board and in case of any discrepancy send a SMS notification as well as a notification on the application developed for the same to the farmer's smartphone using Wi-Fi/3G/4G. The system has a duplex communication link based on a cellular Internet interface that allows for data inspection and irrigation scheduling to be programmed through an android application. Because of its energy autonomy and low cost, the system has the potential to be useful in water limited geographically isolated areas

III. EXISTING SYSTEM

Irrigation automation is already done in the conventional method whereas; the parameters of the irrigation system were continuously uploaded to the server along with the pump on/off control based on the water level in the ground area. Only the required ID is needed to view the graphical analysis of the sensed data. Now this data can be used, by the RMS to monitor the field conditions in real time. In this prototype, the shade and water pump work automatically based on these data. This is a product level approach which can be converted to a fully functional farming system using cheap machineries like the servo motor.

A. *BLOCK DIAGRAM*



Existing block diagram

IV. PROPOSED SYSTEM

HARDWARE REQUREMENTS:

- Central tap transformer
- Rectifier
- Regulator (LM7805)
- Micro controller(16F877A)
- IR Sensor
- Humidity Sensor
- Temperature Sensor
- PH Sensor
- LCD Sensor and IOT module

SOFTWARE REQUIREMENT:

- Proteus 8.1
- CCS C Compiler

PROPOSED SYSTEM EXPLANATION

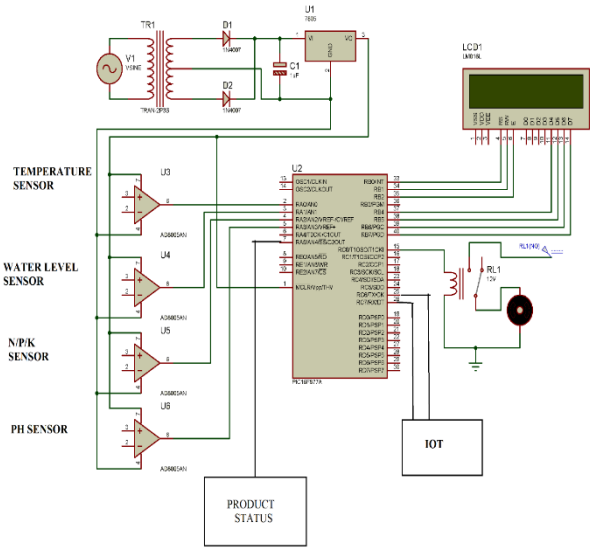
The system first checks the moisture of the soil. If the moisture is less than the threshold value then the irrigation system is checked. If the irrigation system is working normally then the water level at the reservoir is checked. Is the water level is low then the farmer and servicing are informed about the situation and water in the tank is levelled. Analysis has been going on to produce new varieties to increase the yield but if the fields and crops are not monitored properly the results may not be as per expectation. This work invoked to take a preventive measure for temperature, humidity or soil moisture, minerals of soil, crop level, and loss of crop and also increase the productivity of crop. Then the cultivated product and the livestock status were uploaded to the IOT server for the marketing purposes.

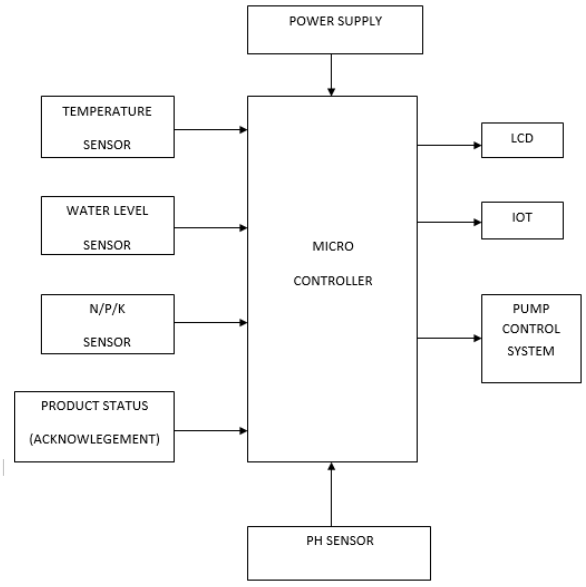
The proposed system mainly focusing the crop identifier based on the ingredients of the soil, and the water availability. Secondly web based marketing for the improvement of the farmers profit level,

V. CIRCUIT DIAGRAM POWER SUPPLY

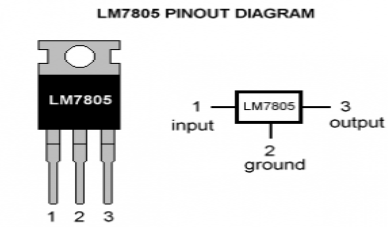
A power supply (sometimes known as a power supply unit or PSU) is a device or system that supplies electrical or other types of energy to an output load or group of loads. The term is most commonly applied to electrical energy supplies, less often to mechanical ones, and rarely to others.

All digital circuits work only with low DC voltage. A power supply unit is required to provide the appropriate voltage supply. This unit consists of transformer, rectifier, filter and a regulator. AC voltage typically of 230Vrms is connected to a transformer which steps that AC voltage down to the desired AC voltage level. A diode rectifier then provides a full wave rectified voltage that is initially filtered by a simple capacitor filter to produce a DC voltage. This resulting DC voltage usually has some ripple or AC voltage variations. Regulator circuit can use this DC input to provide DC voltage that not only has much less ripple voltage but also remains in the same DC value, even when the DC voltage varies, or the load connected to the output DC voltage changes. The required DC supply is obtained from the available AC supply after rectification, filtration and regulation. Block diagram of power supply





The main components used in the power supply unit are Transformer, Rectifier, Filter and Regulator. The 230V AC supply is converted into 9V AC supply through the transformer. The output of the transformer has the same frequency as in the input AC power. This AC power is converted into DC power through diodes. Here the bridge diode is used to convert AC supply to the DC power supply. This converted DC power supply has the ripple content and for normal operation of the circuit, the ripple content of the DC power supply should be as low as possible. Because the ripple content of the power supply will reduce the life of the circuit. So to reduce the ripple content of the DC power supply, the large value of capacitance filter is used.



This filtered output will not be the regulated voltage. For this purpose IC7805 regulator IC is used in the circuit.

TRANSFORMER

Transformer is a device used either for stepping-up or stepping-down the AC supply voltage with a corresponding decreases or increases in the current. Here, a transformer is used for stepping-down the voltage so as to get a voltage that can be regulated **to get a constant 5V.**

RECTIFIER

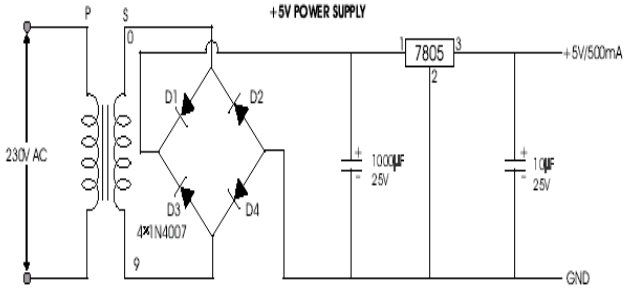
A rectifier is a device like semiconductor, capable of converting sinusoidal input waveform units into a unidirectional waveform, with a nonzero average component.

FILTERS

Capacitors are used as filters in the power supply unit. The action of the system depends upon the fact, that the capacitors stores energy during the conduction period and delivers this energy to the load during the inverse or non-conducting period. In this way, time during which the current passes through the load is prolonged and ripple is considerably reduced.

VOLTAGE REGULATOR

The LM78XX is three terminal regulator available with several fixed output voltages making them



useful in a wide range of applications. IC7805 is a fixed voltage regulators used in this circuit.

REGULATOR 7805

Voltage sources in a circuit may have fluctuations resulting in not providing fixed voltage outputs. A voltage regulator IC maintains the output voltage at a constant value. 7805 IC, a member of 78xx series of fixed linear voltage regulators used to maintain such fluctuations, is a popular voltage regulator integrated circuit (IC). The xx in 78xx indicates the output voltage it provides. 7805 IC provides +5 volts regulated power supply with provisions to add a heat sink. All voltage sources cannot able to give fixed output due to fluctuations in the circuit. For getting constant and steady output, the voltage regulators are implemented. The integrated circuits which are used for the regulation of voltage are termed as voltage regulator ICs. Here, we can discuss about IC 7805.The voltage regulator IC 7805 is actually a member of 78xx series of voltage regulator ICs. It is a fixed linear voltage regulator. The xx present in 78xx represents the value of the fixed output voltage that the particular IC provides. For 7805 IC, it is +5V DC regulated power supply.

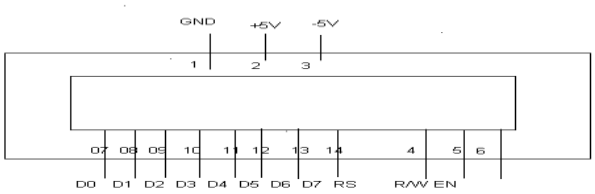
7805 IC Rating:

- Input voltage range 7V- 35V
- Current rating Ic = 1A
- Output voltage range VMax=5.2V ,VMin=4.8V.

The purposes of coupling the components to the IC7805 are explained below. C1- It is the bypass capacitor, used to bypass very small extent spikes to the earth. C2 and C3- They are the filter capacitors. C2 is used to make the slow changes in the input voltage given to the circuit to the steady form. C3 is used to make the slow changes in the output voltage from the regulator in the circuit to the steady form. When the value of these capacitors increases, stabilization is enlarged. But these capacitors single-handedly are unable to filter the very minute changes in the input and output voltages. C4- like C1, it is also a bypass capacitor, used to bypass very small extent spikes to the ground or earth. This is done without influencing other components.

Applications of Voltage Regulator 7805 IC

VI. PIN DIAGRAM OF LCD



PIN DESCRIPTIONS: -

Vcc, Vss and Vee: -

While Vcc and Vss provide +5V and ground respectively, Vee is used for controlling LCD contrast.

RS Register Select: -

There are two very important registers inside the LCD. The RS pin is used for their selection as follows.

If RS=0, the instruction command code register is selected, allowing the user to send a command such as clear display, cursor at home, etc.

If RS=1, the data register is selected, allowing the user to send data to be displayed on the LCD.

R/W, read/write: -

R/W input allows the user to write information to the LCD or read information from it.

R/W = 1 for reading.

R/W= 0 for writing.

EN, enable: -

The LCD to latch information presented to its data pins uses the enable pin. When data is supplied to data pins, a high–to-low pulse must be applied to this pin in order for the LCD to latch in the data present at the data pins. This pulse must be a minimum of 450 ns wide.

D0 – D7: -

The 8–bit data pins, DO – D7, are used to send information to the LCD or read the contents of the LCD's internal registers.

To display letters and numbers, we send ASCII codes for the letters A–Z, a-z numbers 0-9 to these pins while making RS=1.

There are also instruction command codes that can be sent to the LCD to clear the display or force the cursor to home position or blink the instruction command codes.

We also use RS = 0 to check the busy flag bit to see if the LCD is ready to receive information. The busy flag is D7 and can be read when R/W=1 and RS=0, as follows: if R/W = 1, RS = 0. When D7= 1 (busy flag = 1), the LCD is busy taking care of internal operations and will not accept any information.

DIFFERENT ARCHITECTURE

PICmicro chips have a Harvard architecture, and instruction words are unusual sizes. Originally, 12-bit instructions included 5 address bits to specify the memory operand, and 9-bit branch destinations. Later revisions added opcode bits, allowing additional address bits.

Baseline core devices (12 bit)

These devices feature a 12-bit wide code memory, a 32-byte register file, and a tiny two level deep call stack. They are represented by the PIC10 series, as well as by some PIC12 and PIC16 devices. Baseline devices are available in 6-pin to 40-pin packages.

Generally the first 7 to 9 bytes of the register file are special-purpose registers, and the remaining bytes are general purpose RAM. Pointers are implemented using a register pair: after writing an address to the FSR (file select register), the INDF (indirect f) register becomes an alias for the addressed register. If banked RAM is implemented, the bank number is selected by the high 3 bits of the FSR. This affects register numbers 16–31; registers 0–15 are global and not affected by the bank select bits.

Because of the very limited register space (5 bits), 4 rarely read registers were not assigned addresses, but written by special instructions (OPTION and TRIS).

The ROM address space is 512 words (12 bits each), which may be extended to 2048 words by banking. CALL and GOTO instructions specify the low 9 bits of the new code location; additional high-order bits are taken from the status register. Note that a CALL instruction only includes 8 bits of address, and may only specify addresses in the first half of each 512-word page.

The instruction set is as follows. Register numbers are referred to as "f", while constants are referred to as "k". Bit numbers (0–7) are selected by "b". The "d" bit selects the destination: 0 indicates W, while 1 indicates that the result is written back to source register f. The C and Z status flags may be set based on the result; otherwise they are unmodified. Add and subtract (but not rotate) instructions that set C also set the DC (digit carry) flag, the carry from bit 3 to bit 4, which is useful for BCD arithmetic.

ELAN Microelectronics clones (13 bit)

ELAN Microelectronics Corp. make a series of PIC micro-like microcontrollers with a 13-bit instruction word. The instructions are mostly compatible with the mid-range 14-bit instruction set, but limited to a 6-bit register address (16 special-purpose registers and 48 bytes of RAM) and a 10-bit (1024 word) program space.

The 10-bit program counter is accessible as R2. Reads access only the low bits, and writes clear the high bits. An exception is the TBL instruction, which modifies the low byte while preserving bits 8 and 9.

The 7 accumulator-immediate instructions are renumbered relative to the 14-bit PIC micro, to fit into 3 op code bits rather than 4, but they are all there, as well as an additional software interrupt instruction.

There are a few additional miscellaneous instructions, and there are some changes to the terminology (the PICmicro OPTION register is called the CONTrol register; the PICmicro TRIS registers 1–3 are called I/O control registers 5–7), but the equivalents are obvious.

Mid-range core devices (14 bit)

These devices feature a 14-bit wide code memory, and an improved 8 level deep call stack. The instruction set differs very little from the baseline devices, but the 2 additional opcode bits allow 128 registers and 2048 words of code to be directly addressed. There are a few additional miscellaneous instructions, and two additional 8-bit literal instructions, add and subtract. The mid-range core is available in the majority of devices labeled PIC12 and PIC16.

The first 32 bytes of the register space are allocated to special-purpose registers; the remaining 96 bytes are used for general-purpose RAM. If banked RAM is used, the high 16 registers (0x70–0x7F) are global, as are a few of the most important special-purpose registers, including the STATUS register which holds the RAM

bank select bits. (The other global registers are FSR and INDF, the low 8 bits of the program counter PCL, the PC high preload register PCLATH, and the master interrupt control register INTCON.)

The PCLATH register supplies high-order instruction address bits when the 8 bits supplied by a write to the PCL register, or the 11 bits supplied by a GOTO or CALL instruction, is not sufficient to address the available ROM space.

PIC17 high end core devices (16 bit)

The 17 series never became popular and has been superseded by the PIC18 architecture. It is not recommended for new designs, and availability may be limited.

Improvements over earlier cores are 16-bit wide opcodes (allowing many new instructions), and a 16 level deep call stack. PIC17 devices were produced in packages from 40 to 68 pins.

The 17 series introduced a number of important new features:

- a memory mapped accumulator
- read access to code memory (table reads)
- Direct register to register moves (prior cores needed to move registers through the accumulator)
- An external program memory interface to expand the code space
- An 8-bit × 8-bit hardware multiplier
- A second indirect register pair
- Auto-increment/decrement addressing controlled by control bits in a status register (ALUSTA)

PIC18 high end core devices (8 bit)

In 2000, Microchip introduced the PIC18 architecture. Unlike the 17 series, it has proven to be very popular, with a large number of device variants presently in manufacture. In contrast to earlier devices, which were more often than not programmed in assembly, C has become the predominant development language.

The 18 series inherits most of the features and instructions of the 17 series, while adding a number of important new features:

- Call stack is 21 bits wide and much deeper (31 levels deep)
- The call stack may be read and written (TOSU:TOSH:TOSL registers)
- Conditional branch instructions
- Indexed addressing mode (PLUSW)
- extending the FSR registers to 12 bits, allowing them to linearly address the entire data address space
- The addition of another FSR register (bringing the number up to 3)

The RAM space is 12 bits, addressed using a 4-bit bank select register and an 8-bit offset in each instruction. An additional "access" bit in each instruction selects between bank 0 (a=0) and the bank selected by the BSR (a=1).

A 1-level stack is also available for the STATUS, WREG and BSR registers. They are saved on every interrupt, and may be restored on return. If interrupts are disabled, they may also be used on subroutine call/return by setting the s bit (appending ", FAST" to the instruction).

The auto increment/decrement feature was improved by removing the control bits and adding four new indirect registers per FSR. Depending on which indirect file register is being accessed it is possible to post decrement, post increment, or pre increment FSR; or form the effective address by adding W to FSR.

In more advanced PIC18 devices, an "extended mode" is available which makes the addressing even more favorable to compiled code:

- A new offset addressing mode; some addresses which were relative to the access bank are now interpreted relative to the FSR2 register
- The addition of several new instructions, notable for manipulating the FSR registers.

These changes were primarily aimed at improving the efficiency of a data stack implementation. If FSR2 is used either as the stack pointer or frame pointer, stack items may be easily indexed—allowing more efficient re-entrant code. Microchip's MPLAB C18 C compiler chooses to use FSR2 as a frame pointer.

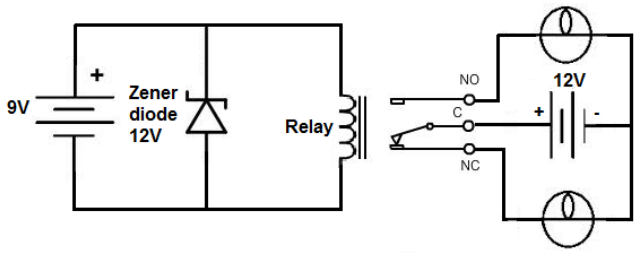
PIC32 32-bit microcontrollers

In November 2007, Microchip introduced the new PIC32MX family of 32-bit microcontrollers. The initial device line-up is based on the industry standard MIPS32 M4K Core. The device can be programmed using the Microchip MPLAB C Compiler for PIC32 MCUs, a variant of the GCC compiler.

The PIC32 architecture brings a number of new features to Microchip portfolio, including:

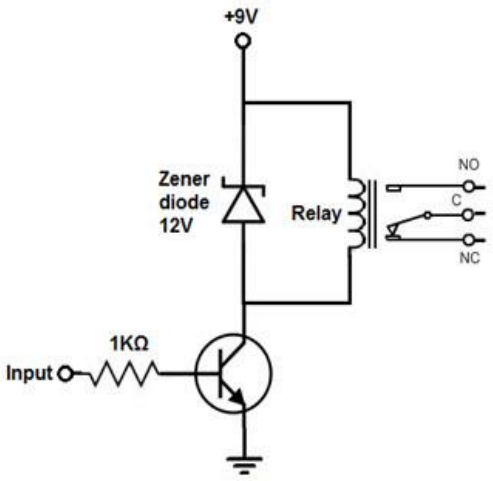
- The highest execution speed 80 MIPS (120+[21] Dhrystone MIPS @ 80 MHz)
- The largest flash memory: 512 KB
- One instruction per clock cycle execution
- The first cached processor
- Allows execution from RAM
- Full Speed Host/Dual Role and OTG USB capabilities
- Full JTAG and 2 wire programming and debugging
- Real-time trace

VII. DC Relay Driver Circuit Schematic



VIII. Generic Relay Driver Circuit

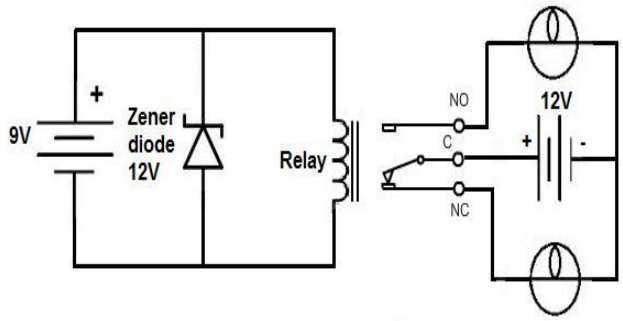
The last relay driver circuit we will show is one which can be driven by an arbitrary control voltage. This is a relay driver circuit which can be driven by either AC or DC input voltage. And unlike the other circuits, a specific voltage, such as the rated voltage values we used to drive the others, does not need to be used. Because this circuit contains a transistor, much less power needs to used on the input side to drive it.



IX. DC Relay Driver IC Circuit

Let us see construction of relay driver circuit for relays that are operated from DC power. In order to drive a DC relay, DC voltage is needed in required quantity to rate a relay and a zener diode. Voltage is required for the relay to operate and to open or close its switch in a circuit. Relays exist with a voltage rating. This is known as relay's datasheet to rate its coil voltage. For the function of relay, it must receive this voltage at its coil terminals. Thus, if a relay has a rated voltage of 9VDC, it should get 9 volts of DC voltage for its working.

In order to eliminate voltage spikes from a relay circuit, a diode is required for its proper functioning. The coil of a relay acts an Inductor.



X. OPERATING PRINCIPLES

In order to connect an object to the IoT, several things are needed in the hardware and software realm. First of all, if one wishes to go beyond simply connecting data from a computer, objects to gather (sensors) or receive (actuators) data are necessary. For example, a digital thermometer can be used to measure temperature. In this case, the data needs to be uploaded to a network of connected servers

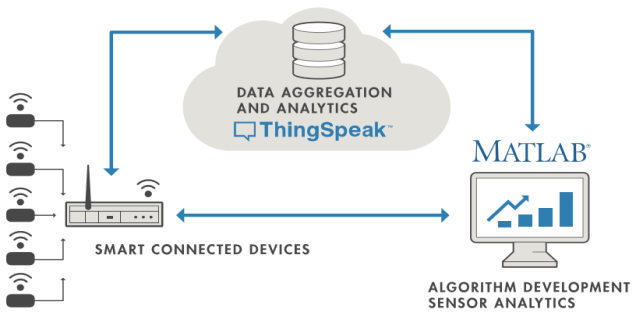
which run applications. Such a network is commonly referred to as 'the cloud'.

The cloud utilizes the process of visualization, meaning that several physical servers can be connected and used in tandem, but appear to the user as one machine (despite that at the physical level, the machines function independently). This method of computing thus allows changes to be made to the 'virtual' server (such as software updates or changes in storage space) much easier than before.

In this case, an object will connect to the cloud through a (possibly wireless) Internet connection to upload or receive data. Objects to be connected are typically augmented with either sensors or actuators.

A sensor is something that tells us about our environment. Think of a temperature sensor, or even the GPS receiver on your mobile phone. Actuators are something that you want to control. Things like thermostats, lights, pumps, and outlets. The IoT brings everything together and allows us to interact with our things and, even more interestingly, allows things to interact with other things. For the purpose of connecting an object to the IoT, we focus on the Thing Speak API. The interface provides simple communication capabilities to objects within the IoT environment, as well as interesting additional applications (such as Thing Tweet, which will be further discussed in a later section). Moreover, Thing Speak allows you to build applications around data collected by sensors. It offers near real-time data collection, data processing, and also simple visualizations for its users.

Data is stored in so-called channels, which provides the user with a list of features. Each channel allows you to store up to 8 fields of data, using up to 255 alphanumeric characters each. There are also 4 dedicated fields for positional data, consisting of: Description, Latitude, Longitude, and Elevation. All incoming data is time and date stamped and receives a sequential ID. Once a channel has been created, data can be published by accessing the ThingSpeak API with a 'write key', a randomly created unique alphanumeric string used for authentication. Consequently, a 'read key' is used to access channel data in case it is set to keep its data private (the default setting). Channels can also be made public in which case no read key is required.



XI. SOFTWARE REQUIREMENT:

MBEEDDED C :

Embedded C is one of the most popular and most commonly used Programming Languages in the development of Embedded Systems. So, in this article, we will see some of the Basics of Embedded C Program and the Programming Structure of Embedded C.

Embedded C is perhaps the most popular languages among Embedded Programmers for programming Embedded Systems. There are many popular programming languages like Assembly, BASIC, C++ etc. that are often used for developing Embedded Systems but Embedded C remains popular due to its efficiency, less development time and portability.

Before digging in to the basics of Embedded C Program, we will first take a look at what an Embedded System is and the importance of Programming Language in Embedded Systems.

EMBEDDED SYSTEM:

An Embedded System can be best described as a system which has both the hardware and software and is designed to do a specific task. A good example for an Embedded System, which many households have, is a Washing Machine We use washing machines almost daily but wouldn't get the idea that it is an embedded system consisting of a Processor (and other hardware as well) and software.

Embedded Systems can not only be stand-alone devices like Washing Machines but also be a part of a much larger system. An example for this is a Car. A modern day Car has several individual embedded systems that perform their specific tasks with the aim of making a smooth and safe. Some of the embedded systems in a Car

are Anti-lock Braking System (ABS), Temperature Monitoring System, Automatic Climate Control, Tyre Pressure Monitoring System, Engine Oil Level Monitor, etc.

XII. PROGRAMMING IN EMBEDDED SYSTEM:

As mentioned earlier, Embedded Systems consists of both Hardware and Software. If we consider a simple Embedded System, the main Hardware Module is the Processor. The Processor is the heart of the Embedded System and it can be anything like a Microprocessor, Microcontroller, DSP, CPLD (Complex Programmable Logic Device) and FPGA (Field Programmable Gated Array).

All these devices have one thing in common: they are programmable i.e. we can write a program (which is the software part of the Embedded System) to define how the device actually works.

Embedded Software or Program allow Hardware to monitor external events (Inputs) and control external devices (Outputs) accordingly. During this process, the program for an Embedded System may have to directly manipulate the internal architecture of the Embedded Hardware (usually the processor) such as Timers, Serial Communications Interface, Interrupt Handling, and I/O Ports etc.

From the above statement, it is clear that the Software part of an Embedded System is equally important to the Hardware part. There is no point in having advanced Hardware Components with poorly written programs (Software).

There are many programming languages that are used for Embedded Systems like Assembly (low-level Programming Language), C, C++, JAVA (high-level programming languages), Visual Basic, JAVA Script (Application level Programming Languages), etc.

In the process of making a better embedded system, the programming of the system plays a vital role and hence, the selection of the Programming Language is very important.

XIII. INTRODUCTION TO EMBEDDED C PROGRAMMING LANGUAGE

Before going in to the details of Embedded C Programming Language and basics of Embedded C Program, we will first talk about the C Programming Language.

The C Programming Language, developed by Dennis Ritchie in the late 60's and early 70's, is the most popular and widely used programming language. The C Programming Language provided low level memory access using an uncomplicated compiler (a software that converts programs to machine code) and achieved efficient mapping to machine instructions.

The C Programming Language became so popular that it is used in a wide range of applications ranging from Embedded Systems to Super Computers.

Embedded C Programming Language, which is widely used in the development of Embedded Systems, is an extension of C Program Language. The Embedded C Programming Language uses the same syntax and semantics of the C Programming Language like main function, declaration of datatypes, defining variables, loops, functions, statements, etc.

The extension in Embedded C from standard C Programming Language include I/O Hardware Addressing, fixed point arithmetic operations, accessing address spaces, etc.

XIV. DIFFERENCE BETWEEN C AND EMBEDDED C

There is actually not much difference between C and Embedded C apart from few extensions and the operating environment. Both C and Embedded C are ISO Standards that have almost same syntax, datatypes, functions, etc.

Embedded C is basically an extension to the Standard C Programming Language with additional features like Addressing I/O, multiple memory addressing and fixed-point arithmetic, etc.

C Programming Language is generally used for developing desktop applications whereas Embedded C is used in the development of Microcontroller based applications.

XV. EMBEDDED SYSTEM AND ITS REAL TIME APPLICATIONS

The World is filled with Embedded Systems. The development of Microcontroller has paved path for several Embedded System application and they play a significant role (and will continue to play in the future as well) in our modern day life in one way or the other.

Starting from consumer electronics like Digital Cameras, DVD Players to high end and advanced systems like Flight Controllers and Missile Guidance Systems, embedded systems are omnipresent and became an important part of our life.

The way we live our life has been significantly improved with the utilization of Embedded Systems and they will continue to be an integral part of our lives even tomorrow.

Another important concept we are hearing these days is Real – Time Systems. In a real time system, Real Time Computing takes place, where a computer (an Embedded System) must generate response to events within certain time limits.

Before going in to the details of Real Time Applications of Embedded Systems, we will first see what an Embedded System is, what is a real time system and what is real time operating system.

Some Other Real Time Applications of Embedded Systems

- Latest Smart TVs
- GPS Navigation Systems
- Almost all Modern Day Smart Phones
- Missile Guidance Systems
- Space Exploration (Rovers)
- Automobiles (ABS, Airbags)
- Industries (Assembly Robots)
- Road Safety Systems (Traffic Monitoring and Collision Alert Systems)
- and many other.

XVI. PIC C COMPILER:

CCS provides a method to attempt to make sure you can compile code written in older versions of CCS with minimal difficulty by altering the methodology to best match the desired version. Currently, there are 4 levels of compatibility provided: CCS V2.XXX, CCS V3.XXX, CCS V4.XXX and ANSI.

Notice: this only affects the compiler methodology, it does not change any drivers, libraries and include files that may have been available in previous versions.

#device CCS2

• ADC default size is set to the resolution of the device (#device ADC=10, #device

ADC=12, etc)

• boolean = int8 is compiled as: boolean = (int8 != 0)

• Overload directive is required if you want to overload functions

• Pointer size was set to only access first bank (PCM *=8, PCB *=5)

• var16 = NegConst8 is compiled as: var16 = NegConst8 & 0xFF (no sign extension)

• Compiler will NOT automatically set certain #fuses based upon certain code conditions.

• rom qualifier is called _rom

#device CCS3

• ADC default is 8 bits (#device ADC=8)

• boolean = int8 is compiled as: boolean = (int8 & 1)

• Overload directive is required if you want to overload functions

• Pointer size was set to only access first bank (PCM *=8, PCB *=5)

• var16 = NegConst8 is compiled as: var16 = NegConst8 & 0xFF (no sign extension)

• Compiler will NOT automatically set certain #fuses based upon certain code conditions.

• rom qualifier is called _rom

#device CCS4

• ADC default is 8 bits (#device ADC=8)

• boolean = int8 is compiled as: boolean = (int8 & 1)

• You can overload functions without the overload directive

• If the device has more than one bank of RAM, the default pointer size is now 16

(#device *=16)

• var16 = NegConst8 is will perform the proper sign extension

• Automatic #fuses configuration (see next section)

#device ANSI

• Same as CCS4, but if there are any discrepancies are found that differ with the ANSI

standard then the change will be made to ANSI

• Data is signed by default

• const qualifier is read-only RAM, not placed into program memory (use rom qualifier to

place into program memory)

• Compilation is case sensitive by default

• Constant strings can be passed to functions (#device PASS_STRINGS_IN_RAM)

This C compiler, is fully optimised for use with PIC microcontrollers. Built in functions make coding the software very easy. Based on original K&R, the integrated C development environment gives developers a fast method to produce efficient code from an easily Maintainable high level language.

CAPABILITIES

- Arrays up to 5 subscripts
- Structures and Unions may be nested.
- Custom bit fields (1-8 bits) within structures.
- Enumerated types,
- Constant variables, arrays and strings.
- Full function parameter support (any number).
- Some support for C++ reference parameters.

STANDARD C FUNCTIONS:

· IF, ELSE, WHILE, DO, SWITCH, CASE, FOR, RETURN, GOTO, BREAK, CONTINUE

· !, ~, ++, --, *, /, %, +, -, <<, >>, <. <=, >, >=, ==, !=, &, ^, |, &&, ||, ?:, =, +=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=

· TYPEDEF, STATIC, AUTO, CONST, ENUM, STRUCT, UNION

XVII. CONCLUSION AND FEATURES

CONCLUSION:

IOT based smart agriculture system can prove to be very helpful for farmers since over as well as less irrigation is not good for agriculture. Threshold values for climatic conditions like humidity, temperature, moisture can be fixed based on the environmental conditions of that particular region. The system also senses the invasion of animals which is a primary reason for reduction in crops. This system generates irrigation schedule based on the sensed real time data from field and data from the weather repository. This system can recommend farmer whether or not, is there a need for irrigation. Continuous internet connectivity is required. This can be overcome by extending the system to the farmer directly on his mobile using IOT using mobile web.

FEATURES:

- Built in Libraries for RS232 serial I/O library, I/O, I2C, discrete I/Oand precision delays.
- Integrates with MPLAB and other simulators/emulators for source level debugging.
- Standard Hex file and debug files ensure compatibility with all programmers.
- Formatted Printf allows easy formatting and display in Hex or decimal.
- Efficient function implementation allows call trees deeper than the hardware stack.
- Access to hardware from easy to use C functions, Timers, A/D, E2, SSP, PSP, I2C & more.

- 1,8, and 16 bit types.
- Assembly code may be inserted anywhere in source and may reference C variables.
- Automatic linking handles multiple code pages.
- Inline procedures supported; Linker automatically determines optimum architecture or it can be manually specified.
- Compiler directives determine if tri-state registers are refreshed on every I/O
- Constants (including strings and arrays) are saved in program memory.
- Standard one bit type (Short Int) permits the compiler to generate efficient Bit oriented code.

XVIII. REFERENCES

[1] [1] Costea, I.M. Nemtanu, F.C. Dumitrescu, C. Banu, C.V. Banu,G. S." Monitoring System with Applications in Road Transport". IEEE, (2014)

[2] [2] [Händel. P.Skog, I. "Cell phone Based Measurement Systems for Road Vehicle Traffic Monitoring and Usage-Based Insurance".vol. 8, no. 4. pp. 1238- - 1248. IEEE.(2014)

[3] [3] An. S. Lee. B. Shin, D. "An overview of wise transportation frameworks." In: Computational Intelligenc Communication Systems and Networks (CICSyN), 2011 .Vol.2.pp.300-340.Third International Conference on. IEEE, (2011)

[4] [5] SagarSukode, ShilpaGite,"Vehicle Traffic Congestion Control And Monitoring System in IOT" ISSN 0973-4562 Volume10, Number8(2015) pp. 19513-19523

[5] Tejashri Gadekar1, Priyanka Chavare2, Komal Chipade3& P.S Togrikar4 'Actualizing Intelligent Traffic Control System for Congestion Control, Ambulance Clearance, and Stolen Vehicle Detection."Vol-2, Issue-4, 2016 ISSN: 2454-1362,

[6] Curiac, D.I. Wireless sensor network security enhancement using directional antennas: State of the art and research challenges. Sensors 2016, 16, 488. [Google Scholar] [CrossRef] [Green Version]

[7] George, R.; Mary, T.A.J. Review on directional antenna for wireless sensor network applications. IET Commun. 2020, 14, 715–722. [Google Scholar] [CrossRef]

[8] Arshad, B.; Ogie, R.; Barthelemy, J.; Pradhan, B.; Verstaevel, N.; Perez, P. Computer vision and IoT-based sensors in flood monitoring and mapping: A systematic review. Sensors 2019, 19, 5012. [Google Scholar] [CrossRef] [PubMed] [Green Version]

[9] Andrienko, G.; Andrienko, N.; Chen, W.; Maciejewski, R.; Zhao, Y. Visual analytics of mobility and transportation: State of the art and further research directions. IEEE Trans. Intell. Transp. Syst. 2017, 18, 2232–2249. [Google Scholar] [CrossRef]

[10] Zeng, W.; Lin, C.; Lin, J.; Jiang, J.; Xia, J.; Turkay, C.; Chen, W. Revisiting the modifiable areal unit problem in deep traffic prediction with visual analytics. IEEE Trans. Vis. Comput. Graph. 2020, 27, 839–848. [Google Scholar] [CrossRef] [PubMed]

[11] Wang, X.; Ning, Z.; Wang, L. Offloading in Internet of vehicles: A fog-enabled real-time traffic management system. IEEE Trans. Ind. Inform. 2018, 14, 4568–4578. [Google Scholar] [CrossRef]

[12] Yuan, J.; Chen, C.; Yang, W.; Liu, M.; Xia, J.; Liu, S. A survey of visual analytics techniques for machine learning. Comput. Vis. Media 2020, 7, 3–36. [Google Scholar] [CrossRef]

[13] Sumi, L.; Ranga, V. Intelligent traffic management system for prioritizing emergency vehicles in a smart city. Int. J. Eng. 2018, 31, 278–283. [Google Scholar]

[14] Wang, X.; Ning, Z.; Hu, X.; Wang, L.; Hu, B.; Cheng, J.; Leung, V.C. Optimizing content dissemination for real-time traffic management in large-scale internet of vehicle systems. IEEE Trans. Veh. Technol. 2018, 68, 1093–1105. [Google Scholar] [CrossRef]

[15] Gunda, P.K. Network-Wide Traffic Congestion Visual Analytics: A Case Study for Brisbane Bluetooth MAC Scanner Data. Ph.D. Thesis, Queensland University of Technology, Brisbane City, QLD, Australia, 2021. [Google Scholar]

[16] Lee, C.; Kim, Y.; Jin, S.; Kim, D.; Maciejewski, R.; Ebert, D.; Ko, S. A visual analytics system for exploring, monitoring, and forecasting road traffic

congestion. IEEE Trans. Vis. Comput. Graph. 2019, 26, 3133–3146. [Google Scholar] [CrossRef] [PubMed]

[17] Riveiro, M.; Lebram, M.; Elmer, M. Anomaly detection for road traffic: A visual analytics framework. IEEE Trans. Intell. Transp. Syst. 2017, 18, 2260–2270. [Google Scholar] [CrossRef]

[18] Nallaperuma, D.; Nawaratne, R.; Bandaragoda, T.; Adikari, A.; Nguyen, S.; Kempitiya, T.; De Silva, D.; Alahakoon, D.; Pothuhera, D. Online incremental machine learning platform for big data-driven smart traffic management. IEEE Trans. Intell. Transp. Syst. 2019, 20, 4679–4690. [Google Scholar] [CrossRef]

[19] Adu-Gyamfi, Y. GPU-Enabled Visual Analytics Framework for Big Transportation Datasets. J. Big Data Anal. Transp. 2019, 1, 147–159. [Google Scholar] [CrossRef] [Green Version]

[20] Lock, O.; Bednarz, T.; Pettit, C. The visual analytics of big, open public transport data–a framework and pipeline for monitoring system performance in Greater Sydney. Big Earth Data 2021, 5, 134–159. [Google Scholar] [CrossRef]

[21] De Souza, A.M.; Brennand, C.A.; Yokoyama, R.S.; Donato, E.A.; Madeira, E.R.; Villas, L.A. Traffic management systems: A classification, review, challenges, and future perspectives. Int. J. Distrib. Sens. Netw. 2017, 13, 1550147716683612. [Google Scholar] [CrossRef]