L
A
B

M
A
N
U
A
L

EASWARI ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

191ITC612L- EMBEDDED SYSTEMS AND IOT LABORATORY

LAB MANUAL

Regulations – R19- V21

III YEAR / VI SEM

ACADEMIC YEAR: 2023-2024 EVEN

PREPARED BY

APPROVED BY

FACULTY IN CHARGE

HOD

EASWARI ENGINEERING COLLEGE

DEPARTMENT OF INFORMATION TECHNOLOGY

Vision

- To impart Quality Education towards the holistic development of students and be a strategic partner in the Industrial Advancement arena and emerge as a 'Center of Excellence for Higher Studies' in the specialization of Information Technology.

Mission

- To offer doctoral programmes in the field of Information Technology to enhance research activities.
- To awaken the young minds and lay solid Engineering foundation among the graduates through the design of experiments, analysis and interpretation of data.
- To produce graduates with ethical principles and commit to professional ethics to cater to the norms of engineering practice.
- To create graduates to work individually and as a member of a team to function effectively in multi-disciplinary areas for solving complex engineering problems.
- To use modern Information Technology tools and appropriate teaching techniques for predicting and modeling the real world problems.
- To provide contextual knowledge among graduates to assess societal, health, safety, legal and cultural issues through innovative professional engineering practice.
- To prepare and build the ability to recognise the need for independent and lifelong learning in the context of technological changes in the field of Information Technology.

Program Educational Objectives (PEO)

- Graduates will be proficient in utilizing the fundamental knowledge of basic sciences, mathematics and Information Technology for the applications relevant to various streams of Engineering and Technology.
- Graduates will possess core competencies necessary for applying knowledge of computers and telecommunications equipment to store, retrieve, transmit, manipulate and analyze data in the context of business enterprise.
- Graduates will be capable of thinking logically, pursue lifelong learning and will have the

capacity to understand technical issues related to computing systems and to design optimal solutions.

- Graduates will be able to develop hardware and software systems by understanding the importance of social, business and environmental needs in the human context.
- Graduates will gain employment in organizations and establish themselves as professionals by applying their technical skills to solve real world problems and meet the diversified needs of industry, academia and research.

Program Outcomes (PO)

Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

Problem analysis: Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change

Program Specific Outcomes (PSO)

- Create, select, and apply appropriate techniques, resources, modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.
- Manage complex IT projects with consideration of the human, financial, ethical and environmental factors and an understanding of risk management processes, and operational and policy implications.

191ITC612L	EMBEDDED SYSTEMS AND IOT LABORATORY	Periods per week				Credits
		L	T	P	R	
		0	0	3	1	2

PREREQUISITES:
NIL

COURSE OBJECTIVES:

1.	Learn the working of ARM processor
2.	Understand the Building Blocks of Embedded Systems
3.	Learn the concept of memory map and memory interface
4.	Know the characteristics of Real Time Systems
5.	Write programs to interface memory, I/O s with processor.
6.	Study the performance of interrupts.

LIST OF PROGRAMS:

1.	Study of PIC microcontroller system.
2.	Study of ARM microcontroller system
3.	Toggle all the led to port and with some time delay using ARM7
4.	Interfacing analog to digital converter with ARM processor
5.	Interface LED and PWM and to verify the output in the ARM7
6.	Interfacing Real Time clock and Serial port.
7.	Interfacing Keyboard and LCD.
8.	Interrupt performance characteristics of ARM and FPGA.
9.	Interfacing Stepper motor and Temperature sensor.
10.	Turn an LED on and off with Arduino.
11.	Read a switch, print the state out to the Arduino Serial Monitor
12.	Read an analog input and print the voltage to the Arduino Serial Monitor.
13.	Detect knocks with a Piezo element sensor.
14.	Study and implementation of IoT using Arduino/Raspberry pi.
15.	Mini Projects using embedded system and IoT.

TOTAL PERIODS:	**60**

Content Beyond Syllabus(CBS)

- To create a traffic light using an Arduino Uno.

- Arduino Based Home Security System Using PIR Sensor

COURSE OUTCOMES:	
Upon completion of this course, student will be able to:	
CO1:	Design applications using ARM processor.
CO2:	Implement programs for memory interface with ARM processor.
CO3:	Interface A/D and D/A convertors with ARM system.
CO4:	Analyze the impact of interrupts on system performance.
CO5:	Develop programs for hardware interfacing with Arduino.
CO6:	Formulate a mini project using embedded system and IoT.

LIST OF EQUIPMENTS:

1. HARDWARE:

- Embedded trainer kits with ARM board
- Adequate quantities of Hardware, software and consumables.

2. SOFTWARE

- Arduino IDE
- KEIL
- FLASHMAGIC

COURSE OUTCOMES

191ITC612L.1	Design applications using ARM processor.
191ITC612L.2	Implement programs for memory interface with ARM processor.
191ITC612L.3	Interface A/D and D/A convertors with ARM system.
191ITC612L.4	Analyze the impact of interrupts on system performance.
191ITC612L.5	Develop programs for hardware interfacing with Arduino.
191ITC612L.6	Formulate a mini project using embedded system and IoT

CO-PO-PSO MATRIX

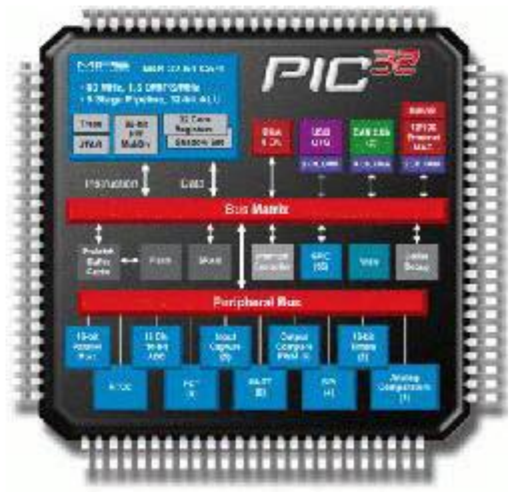
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
191ITC612L.1	3	3	3	3	3	-	-	2	3	-	2	2	3	1
191ITC612L.2	3	3	3	3	3	-	-	2	3	-	2	2	3	1
191ITC612L.3	3	3	3	3	3	-	-	2	3	-	2	2	3	1
191ITC612L.4	3	3	3	3	3	-	-	2	3	-	2	2	3	2
191ITC612L.5	3	3	3	3	3	-	-	2	3	-	2	2	3	2
191ITC612L.6	3	3	3	3	3	3	3	3	3	-	2	2	3	3

JUSTIFICATION OF THE MAPPING:

191ITC612L.1	Fundamental knowledge of mathematics and engineering is required to understand thebasicsofprogramming(PO1).ProgramminginMicroprocessorcanbeformulated andanalyzedwiththefundamentalslearnt(PO2).Itinculcatestheabilitytodevelop solutionsforrealtimeapplications(PO3)throughexperimentation(PO4)andusageof microprocessor kits.(PO5).Itinculcates the ability to develop solutions for real time applications (PSO1) through experimentation (PSO3) and usage of microprocessor to provide solution to new ideas and innovations. (PSO4).
191ITC612L.2	Agoodmathematicalandengineeringfundamentalisrequiredtowriteprogramsto interface microprocessor with I/O devices (PO1) and hence some knowledge is requiredtoformulateproblems(PO2).Interfacingtorealtimeapplicationsisdoneby conductingexperiments(PO4)andtherebydevelopsolutionstocatertothesocietal needs(PO3).Usageoftools(PO5)isneededtointerfacetheinputandoutputdevices.Itinculcates the ability to develop solutions for real time applications (PSO1) through experimentation (PSO3) and usage of microprocessor to provide solution to new ideas and innovations. (PSO4).
191ITC612L.3	Competent analytical and engineering knowledge is needed to perform interfacing (PO1). Problems encountered during interfacing of analog to digital converter is identified in the literature(PO2) is analyzed with modern tools(PO5)and solutionsaredeveloped(PO3)throughexperimentation(PO4).Itinculcates the ability to develop solutions for real time applications (PSO1) through experimentation (PSO3) and usage of microprocessor to provide solution to new ideas and innovations. (PSO4).
191ITC612L.4	Anin-depthknowledgeofmathematicsandengineeringisrequiredtounderstandthe basics of programming(PO1).Programming in Microcontroller can be formulated and analyzed with the fundamentals learnt (PO2). It inculcates the ability to develop solutions for real time applications(PO3)through experimentation(PO4)and usage of microcontroller kits.(PO5).It inculcates the ability to develop solutions for real time applications (PSO1) through experimentation (PSO3) and usage of microprocessor to provide solution to new ideas and innovations. (PSO4).
191ITC612L.5	Agoodmathematicalandengineeringfundamentalisrequiredtowriteprogramsto interface microcontroller with I/O devices (PO1) and hence some knowledge is required to formulate problems interfacing(PO2). Interfacing to real-time applications is done by conducting experiments (PO4) and thereby develop solutions to cater to the societal needs (PO3). Usage of tools (PO5) is needed to interface the input and output devices. It inculcates the ability to develop solutions for real time applications (PSO1) through experimentation (PSO3) and usage of microprocessor to provide solution to new ideas and innovations. (PSO4).
191ITC612L.6	Competent analytical and engineering knowledge is needed to build IoT projects (PO1). Problems encountered during interfacing of analog to digital converter is identified in the literature(PO2)is analyzed with modern tools(PO5)and solutionsaredeveloped(PO3)throughexperimentation(PO4).Itinculcates the ability to develop solutions for real time applications (PSO1) through experimentation (PSO3) and usage of microprocessor to provide solution to new ideas and innovations. (PSO4).

Ex.No : 1 STUDY OF PIC MICROCONTROLLER SYSTEM

Introduction:



PIC microcontroller was developed in the year 1993 by microchip technology. The term PIC stands for Peripheral Interface Controller. Initially this was developed for supporting PDP computers to control its peripheral devices, and therefore, named as a peripheral interface device. These microcontrollers are very fast and easy to execute a program compared with other microcontrollers. PIC Microcontroller architecture is based on Harvard architecture. PIC microcontrollers are very popular due to their ease of programming, wide availability, easy to interfacing with other peripherals, low cost, large user base and serial programming capability (reprogramming with flash memory), etc.

We know that the microcontroller is an integrated chip which consists of CPU, RAM, ROM, timers, and counters, etc. In the same way, PIC microcontroller architecture consists of RAM, ROM, CPU, timers, counters and supports the protocols such as SPI, CAN, and UART for interfacing with other peripherals. At present PIC microcontrollers are extensively used for industrial purpose due to low power consumption, high performance ability and easy of availability of its supporting hardware and software tools like compilers, debuggers and simulators.

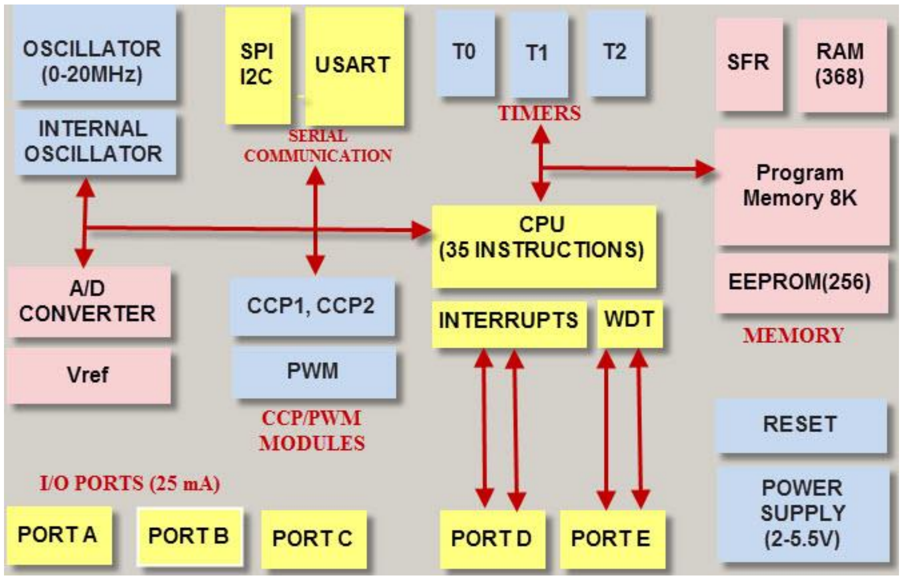
What is a PIC Microcontroller?

PIC (Programmable Interface Controllers) microcontrollers are the worlds smallest microcontrollers that can be programmed to carry out a huge range of tasks. These microcontrollers are found in many electronic devices such as phones, computer control systems, alarm systems, embedded systems, etc. Various types of microcontrollers exist, even though the best are found in the GENIE range of programmable microcontrollers. These microcontrollers are programmed and simulated by a circuit-wizard software.

Every PIC microcontroller architecture consists of some registers and stack where registers function as Random Access Memory(RAM) and stack saves the return addresses. The main features of PIC microcontrollers are RAM, flash memory, Timers/Counters, EEPROM, I/O Ports, USART, CCP (Capture/Compare/PWM module), SSP, Comparator, ADC (analog to digital converter), PSP(parallel slave port), LCD and ICSP (in circuit serial programming) The 8-bit PIC microcontroller is classified into four types on the basis of internal architecture such as Base Line PIC, Mid Range PIC, Enhanced Mid Range PIC and PIC18

Architecture of PIC Microcontroller

The PIC microcontroller architecture comprises of CPU, I/O ports, memory organization, A/D converter, timers/counters, interrupts, serial communication, oscillator and CCP module which are discussed in detailed below.



Architecture of PIC Microcontroller

CPU (Central Processing Unit)

It is not different from other microcontrollers CPU and the PIC microcontroller CPU consists of the ALU, CU, MU and accumulator, etc. Arithmetic logic unit is mainly used for arithmetic operations and to take logical decisions. Memory is used for storing the instructions after processing. To control the internal and external peripherals, control unit is used which are connected to the CPU and the accumulator is used for storing the results and further process.

Memory Organization

The memory module in the PIC microcontroller architecture consists of RAM (Random Access Memory), ROM (Read Only Memory) and STACK.

Random Access Memory (RAM)

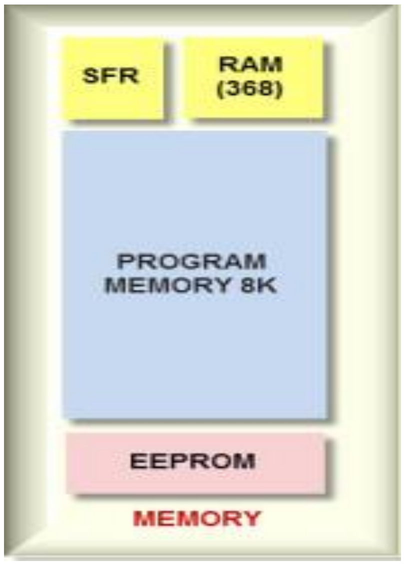
RAM is an unstable memory which is used to store the data temporarily in its registers. The RAM memory is classified into two banks, and each bank consists of so many registers. The RAM registers are classified into two types: Special Function Registers (SFR) and General Purpose Registers (GPR).

- **General Purpose Registers (GPR)**

These registers are used for general purpose only as the name implies. For example, if we want to multiply two numbers by using the PIC microcontroller. Generally, we use registers for multiplying and storing the numbers in other registers. So these registers don't have any special function,- CPU can easily access the data in the registers.

- **Special Function Registers**

These registers are used for special purposes only as the name SFR implies. These registers will perform according to the functions assigned to them , and they cannot be used as normal registers. For example, if you cannot use the STATUS register for storing the data, these registers are used for showing the operation or status of the program. So, user cannot change the function of the SFR; the function is given by the retailer at the time of manufacturing.



Memory Organization

Read Only Memory (ROM)

Read only memory is a stable memory which is used to store the data permanently. In PIC microcontroller architecture, the architecture ROM stores the instructions or program, according to the program the microcontroller acts. The ROM is also called as program memory, wherein the user will write the program for microcontroller and saves it permanently, and finally the program is executed by the CPU. The microcontrollers performance depends on the instruction, which is executed by the CPU.

Electrically Erasable Programmable Read Only Memory (EEPROM)

In the normal ROM, we can write the program for only once we cannot use again the microcontroller for multiple times. But, in the EEPROM, we can program the ROM multiple times.

Flash Memory

Flash memory is also programmable read only memory (PROM) in which we can read, write and erase the program thousands of times. Generally, the PIC microcontroller uses this type of ROM.

Stack

When an interrupt occurs, first the PIC microcontroller has to execute the interrupt and the existing process address. Then that is being executed is stored in the stack. After completing the execution of the interrupt, the microcontroller calls the process with the help of address, which is stored in the stack and get executes the process.

I/O Ports

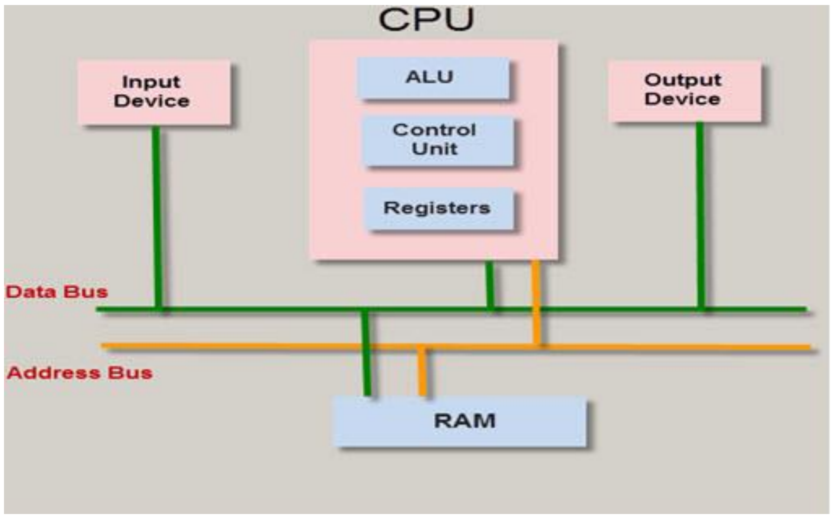
- The series of PIC16 consists of five ports such as Port A, Port B, Port C, Port D & Port E.
- Port A is an 16-bit port that can be used as input or output port based on the status of the TRISA (Tradoc Intelligence Support Activity) register.
- Port B is an 8- bit port that can be used as both input and output port.
- Port C is an 8-bit and the input of output operation is decided by the status of the TRISC register.
- Port D is an 8-bit port acts as a slave port for connection to the microprocessor BUS.
- Port E is a 3-bit port which serves the additional function of the control signals to the analog to digital converter.

BUS

BUS is used to transfer and receive the data from one peripheral to another. It is classified into two types such as data bus and address.

Data Bus: It is used for only transfer or receive the data.

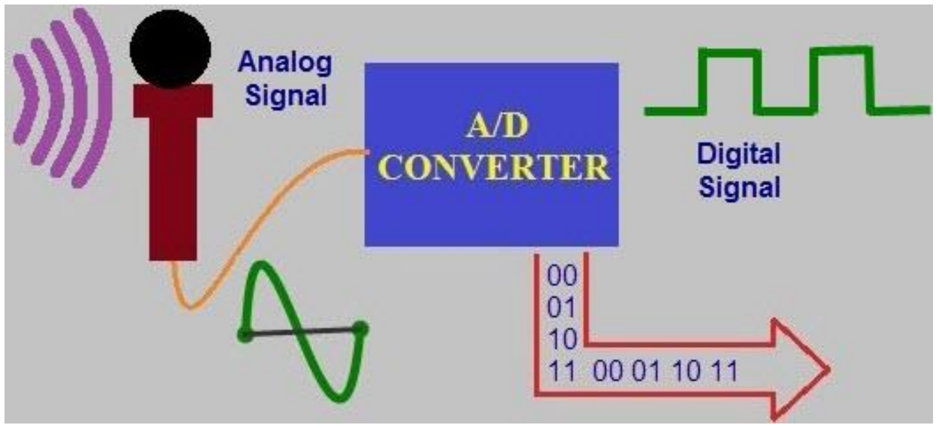
Address Bus: Address bus is used to transmit the memory address from the peripherals to the CPU. I/O pins are used to interface the external peripherals; UART and USART both are serial communication protocols which are used for interfacing serial devices like GSM, GPS, Bluetooth, IR , etc.



BUS

A/D converters

The main intention of this analog to digital converter is to convert analog voltage values to digital voltage values. A/D module of PIC microcontroller consists of 5 inputs for 28 pin devices and 8 inputs for 40 pin devices. The operation of the analog to digital converter is controlled by ADCON0 and ADCON1 special registers. The upper bits of the converter are stored in register ADRESH and lower bits of the converter are stored in register ADRESL. For this operation, it requires 5V of an analog reference voltage.



A/D CONVERTER

Timers/ Counters

PIC microcontroller has four timer/counters wherein the one 8-bit timer and the remaining timers have the choice to select 8 or 16-bit mode. Timers are used for generating accuracy actions, for example, creating specific time delays between two operations.

Interrupts

PIC microcontroller consists of 20 internal interrupts and three external interrupt sources which are associated with different peripherals like ADC, USART, Timers, and so on.

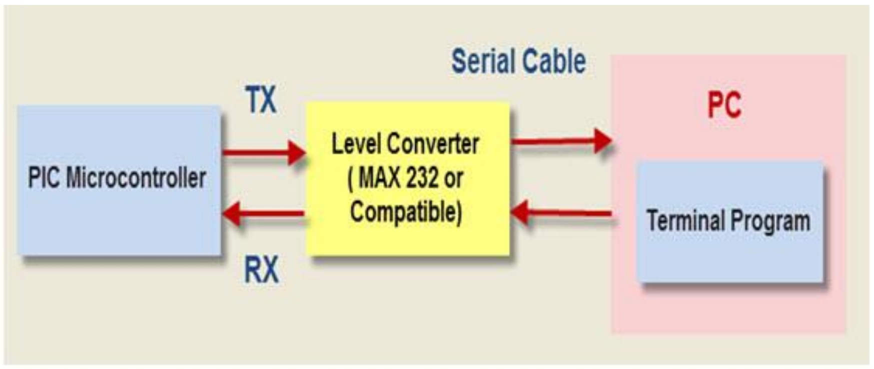
Serial Communication

Serial communication is the method of transferring data one bit at a time sequentially over a communication channel.

- **USART:** The name USART stands for Universal synchronous and Asynchronous Receiver and Transmitter which is a serial communication for two protocols. It is used for transmitting and receiving the data bit by bit over a single wire with respect to clock pulses. The PIC microcontroller has two pins TXD and RXD. These pins are used for transmitting and receiving the data serially.

- **SPI Protocol:** The term SPI stands for Serial Peripheral Interface. This protocol is used to send data between PIC microcontroller and other peripherals such as SD cards, sensors and shift registers. PIC microcontroller support three wire SPI communications between two devices on a common clock source. The data rate of SPI protocol is more than that of the USART.

- **I2C Protocol:** The term I2C stands for Inter Integrated Circuit , and it is a serial protocol which is used to connect low speed devices such as EEPROMS, microcontrollers, A/D converters, etc. PIC microcontroller support two wire Interface or I2C communication between two devices which can work as both Master and Slave device.



Serial Communication

Oscillators

Oscillators are used for timing generation. Pic microcontroller consist of external oscillators like RC oscillators or crystal oscillators. Where the crystal oscillator is connected between the two oscillator pins. The value of the capacitor is connected to every pin that decides the mode of the operation of the oscillator. The modes are crystal mode, high-speed mode and the low-power mode. In case of RC oscillators, the value of the resistor & capacitor determine the clock frequency and the range of clock frequency is 30KHz to 4MHz.

CCP module

The name CCP module stands for capture/compare/PWM where it works in three modes such as capture mode, compare mode and PWM mode.

- **Capture Mode:** Capture mode captures the time of arrival of a signal, or in other words, when the CCP pin goes high, it captures the value of the Timer1.

- **Compare Mode:** Compare mode acts as an analog comparator. When the timer1 value reaches a certain reference value, then it generates an output.

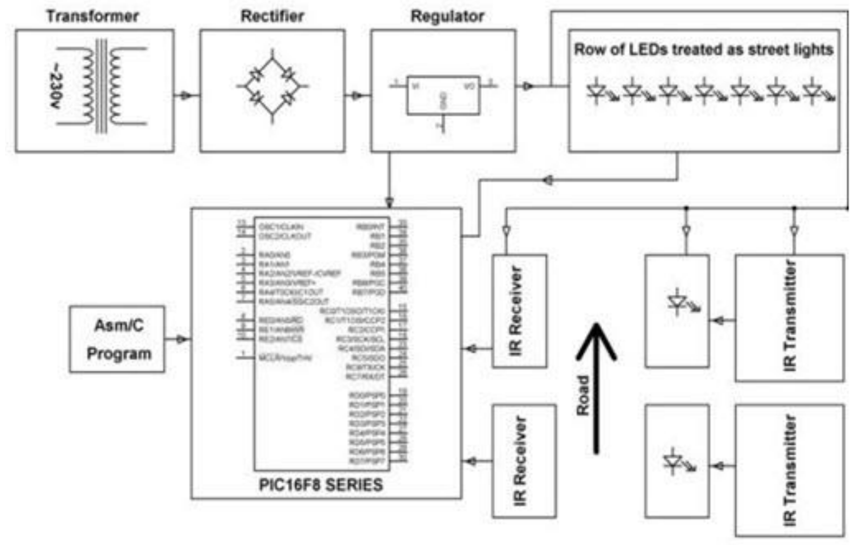
- **PWM Mode:** PWM mode provides pulse width modulated output with a 10-bit resolution and programmable duty cycle.

PIC Microcontroller Applications

The PIC microcontroller projects can be used in different applications, such as peripherals, audio accessories, video games, etc. For better understanding of this PIC microcontroller, the following project demonstrates PIC microcontroller's operations.

Street Light that Glows on Detecting Vehicle Movement:

The main intention of this project is to detect the movement of vehicles on highways to switch on a block of street lights ahead of it, and also switch off the trailing lights to conserve energy. In this project, a PIC microcontroller is done by using assembly language or embedded C.



Street Light that Glows on Detecting Vehicle Movement by Edgefxkits.com

The power supply gives the power to the total circuit by stepping down, rectifying, filtering and regulating AC mains supply. When there are no vehicles on highway, then all lights will turn OFF so that the power can be conserved. The IR sensors are placed on the road to sense the vehicle movement. When there are vehicles on highway, then the IR sensor senses the vehicle movement immediately, it sends the commands to the PIC microcontroller to switch ON/OFF the LEDs. A bunch of LEDS will be turned on when a vehicle come near to the sensor and once the vehicle passes away from the sensor the intensity will become lower than the LEDs will turn OFF

Advantages of PIC Microcontroller:

- PIC microcontrollers are consistent and faulty of PIC percentage is very less. The performance of the PIC microcontroller is very fast because of using RISC architecture.
- When comparing to other microcontrollers, power consumption is very less and programming is also very easy.
- Interfacing of an analog device is easy without any extra circuitry

Disadvantages of PIC Microcontroller:

- The length of the program is high due to using RISC architecture (35 instructions)
- One single accumulator is present and program memory is not accessible

Ex.No : 2 **STUDY OF ARM MICROCONTROLLER SYSTEM**

AIM:

To studythe arm evaluation system lpc4088.

THEORY:

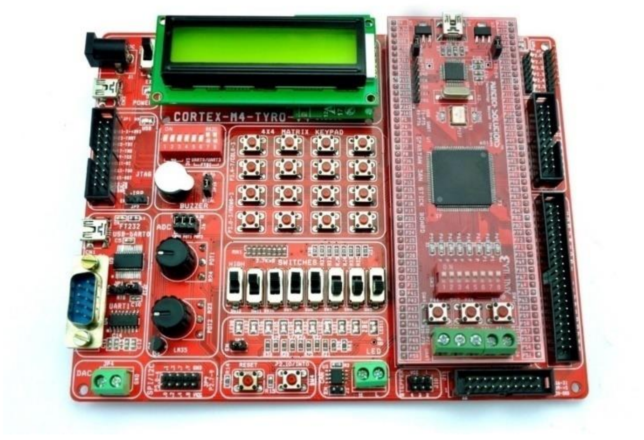
Arm Development board:

The ARM cortex m4 development board has the following hardware features:

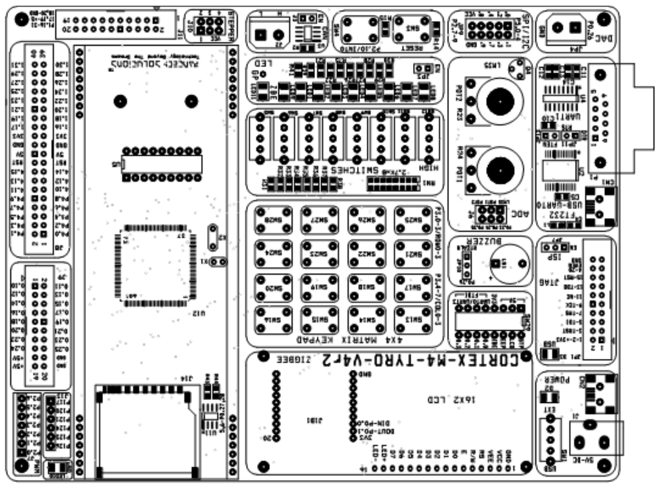
- 8 nos. Point leds (logic output)
- 8 nos. Digital input(slide switch)
- 2 nos. Analog input (potentiometer)
- 2x16 char lcd interface
- Temperature sensor(lm35)
- Internal rtc with battery-backup
- 1 no. Uart(rs232)
- 1 no. Usbuart
- Usb 2.0 device (virtual port)
- DAC output
- Interrupts study, reset button
- 4x4 matrix keyboard
- 40-pin expansion connector
- Jtag (program/debug) |isp programming
- 2 nos. 20pin- i/o expansion connector
- PWM terminations
- Stepper interface
- Optional onboard zigbee interface
- Onboard buzzer
- Optional spartan3an fpga stick interface

- Spi/i2c expansion connector
- Sd card interface
- Can interface (optional)

ARM CORTEX M4 Board



ARM CORTEX BOARD LAYOUT

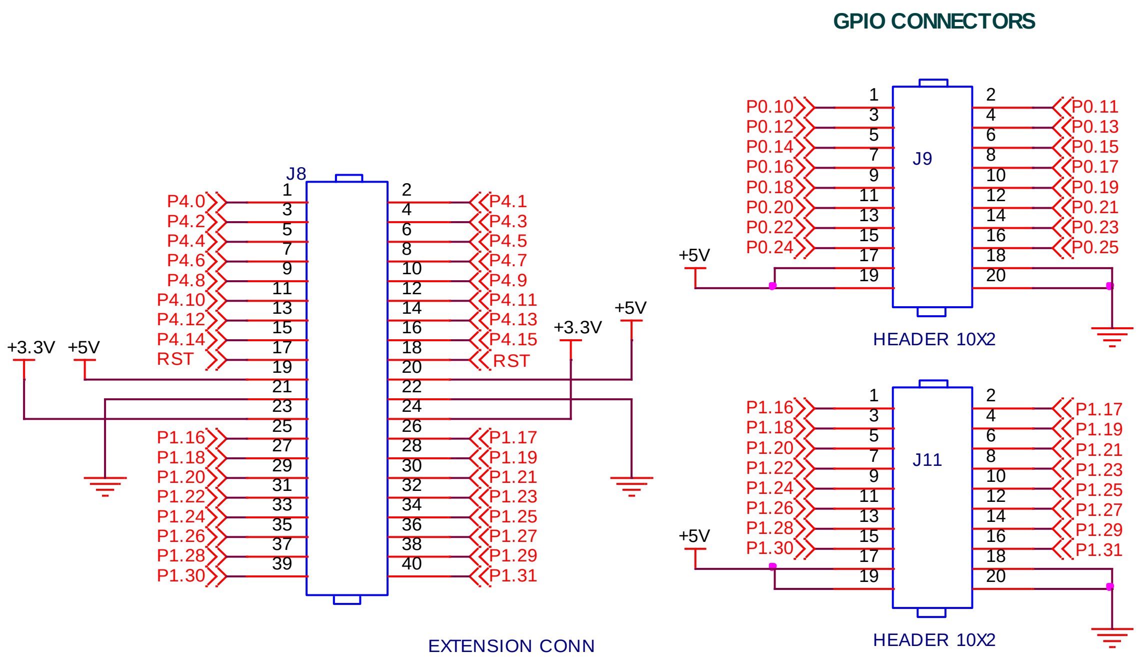


Connectors

40 pin frc box type connector: Instead of terminating each port separately, this connector has all the port pins. So more IO lines can be taken by using single cable.

The ports are arranged as shown in the following figures

Similarly, two number of 20x2 connector gives access to various port lines.



1.1 Features of ARM Microcontroller

- 16-bit/32-bit ARM7TDMI-S microcontroller in a tiny LQFP64 package.
- 8 kB to 40 kB of on-chip static RAM and 32 kB to 512 kB of on-chip flash memory; 128-bit wide interface/accelerator enables high-speed 60 MHz operation.
- In-System Programming/In-Application Programming (ISP/IAP) via on-chip boot loader software, single flash sector or full chip erase in 400 ms and programming of 256 Bytes in 1 ms Embedded ICE RT and Embedded Trace interfaces offer real-time debugging with the on-chip Real Monitor software and high-speed tracing of instruction execution.
- USB 2.0 Full-speed compliant device controller with 2kB of endpoint RAM. In addition, the LPC2148 provides 8 kB of on-chip RAM accessible to USB by DMA.
- One or two (LPC2141/42 vs, LPC2144/46/48) 10-bit ADCs provide a total of

6/14 analog inputs, with conversion times as low as 2.44 ms per channel.

- Single 10-bit DAC provides variable analog output (LPC2148 only)
- Two 32-bit timers/external event counters (with four capture and four compare channels each), PWM unit (six outputs) and watchdog.
- Low power Real-Time Clock (RTC) with independent power and 32 kHz clock input.
- Multiple serial interfaces including two UARTs , two Fast I2C-bus (400 kbit/s),SPI and SSP with buffering and variable data length capabilities.
- Vectored Interrupt Controller (VIC) with configurable priorities and vector addresses.
- Up to 45 of 5 V tolerant fast general purpose I/O pins in a tiny LQFP64 package.
- Up to nine edge or level sensitive external interrupt pins available.
- 60 MHz maximum CPU clock available from programmable on-chip PLL with settling time of 100 ms.
- Power saving modes include Idle and Power-down
- Individual enable/disable of peripheral functions as well as peripheral clock scaling for additional power optimization.
- Processor wake-up from Power-down mode via external interrupt or BOD.
- Single power supply chip with POR and BOD circuits:

CPU operating voltage range of 3.0 V to 3.6 V (3.3 V ± 10 %) with 5 V tolerant I/O.

1.2 Brief overview of ARM7 Architecture

The ARM7TDMI core is a 32-bit embedded RISC processor delivered as a hard macrocell optimized to provide the best combination of performance, power and area characteristics. The ARM7TDMI core enables system designers to build embedded devices requiring small size, low power and high performance. The ARM7 family also includes the ARM7TDMI processor, the ARM7TDMI-S processor, the ARM720T processor and the ARM7EJS processors, each of which has been developed to address different market requirements. The market for microprocessors continues to diversify, based on the evolving demands of applications including wireless, home entertainment, automotive and microcontrollers. ARM core families sharing the ARMv7 architecture will cover the widening spectrum of embedded processing.

The ARM architecture is based on Reduced Instruction Set Computer (RISC) principles. The RISC instruction set and related decode mechanism are much simpler than those of Complex Instruction Set Computer (CISC) design.
This simplicity gives:

- A high instruction throughput
- An excellence real-time interrupts response
- A small, cost-effective, processor macrocell

The ARM7TDMI core is the industry's cost widely used 32-bit embedded RISC microprocessor solution. Optimized for cost and power-sensitive application, the ARM7TDMI solution provides low power consumption, small size, and high performance needed in portable, embedded application.

The ARM7DMI-S is synthesizable version of ARM7TDMI core. The ARM720T hard macrocell contain the ARM7DMI core, 8KB unified cache and MMU (Memory Management Unit) that allows the use of protected execution space and virtual memory.

The ARM7EJ-S processor is synthesizable core that provides all the benefit of ARM7DMI, while also incorporating ARM's latest DSP extensions and jazelle technology, enabling acceleration of Java-based applications.

Architecture:

The ARM7 core is based on the von Neumann architecture with 32-bit data bus that carries both instruction and data. Data can be of 8 bits, 16 bits, 32 bits. It has following features:

- Instruction pipeline
- Memory format
- Operation modes
- Coprocessor
- Debugging feature

Instruction pipeline:

The ARMv7 core uses a three stage pipeline to increase the flow of instructions to the processor. This allows multiple simultaneous operations to take place and continuous operations and memory systems. The instructions are executed in three stages:

- Fetch
- Decode
- Execute

During normal operation, while one instruction is being executed, its successor is being decoded, and third instruction is being fetched from the memory. The program counter (PC) value used in an executing instruction is always two instructions ahead of the address.

Memory Format:

The ARM7 memory interface is design to allow optimum performance potential and minimize memory usage. Speed critical control signals are pipelined to allow system control function to exploit the fast burst access modes supported by many memory technologies. ARM7 has four basics types of cycle:

- Internal
- Non sequential
- Sequential
- Coprocessor transfer

The ARM7 can be configured to store the words as either in little-endian or big-endian format.
The ARM7 processor supports the following data types:

- Word, 32-bit
- Half word, 16-bit
- Byte, 8-bit

You must align this as follow:

- Word quantities must be aligned to four-byte boundaries.
- Half word quantities must be aligned to two-byte boundaries.
- Byte quantities can be placed on any boundary.

The ARM core supports two operating states and instruction sets

- ARM state for 32 bit word aligned instruction
- Thumb state for 16-bit half word aligned instruction

Operating modes:
The ARMv7 core has seven modes of operation:

- User mode – normal ARM program execution mode and used for executing
most application programs.
- Fast Interrupt (FIQ) – mode supports data transfer or channel processes to allow
very fast interrupt
- Interrupt (IRQ) – mode is used for general purpose interrupt handling.
- Supervisor (SVC) – is protected mode for operating system.
- Abort (ABT) – mode is entered after a data or instruction fetch is aborted.
- Undefined (UND) – mode is entered when an undefined instruction is executed.
- System (SYS) – is a privileged user mode for the operating system.

Modes other than user mode are collectively known as privileged modes. Privileged modes are used to service interrupts or exceptions, or to access protected resources.The ARMv7 has 37 register all are 32bit wide, not all the registers are available for a given modes. R15 is program counter. R14 is link register. R13 stack pointer.
CPSR – current program status register.
SPSR – saved program status register.

Coprocessor:
Up to 16 coprocessors can be connected to an ARMv7 system. Coprocessors are separate processing unit that tightly coupled to the ARM processor. Typical coprocessor contains:

· An instruction pipeline
· Instruction decode logic
· Handshake logic
· A register bank

· Special processing logic with its own data path

Debugging Feature:

Internal state of the ARM core can be examined using a JTAG interface to allow the insertion of instructions into core pipeline and avoid using external data bus.ARM7TDMI core includes an internal functional unit known as the Embedded ICE logic. The embedded ICE logic is configured to monitor the ARM7TDMI core actively for specific instruction fetches and data accesses.

Applications:

Using the ARMv7 architecture, ARM can strengthen its position as a low power/performance leader while conquering new markets to carry its cores up in high performance and down in the low-cost high-volume domain of the microcontroller ARM designs the technology that lies at the heart of advanced digital products, from wireless, networking and consumer entertainment solutions to imaging, automotive, security and storage devices. ARM's comprehensive product offering includes 16/32-bit RISC microprocessors, data engines, 3D processors, digital libraries, embedded memories, peripherals, software and development tools, as well as analog functions and high-speed connectivity products.

1.3 ARM operating modes

The processor mode determines which registers are active and the access rights to the CPSR register itself. Each processor mode is either privileged or non privileged; A privileged mode allows full read-write access to the CPSR.

Conversely, a non privileged mode only allows read access to the control field in the CPSR but still allows read-write access to the condition flags. There are seven processor/operating modes in total: six privileged mode

- Abort mode
- Fast interrupt request mode
- Interrupt request mode Privileged
- Supervisor mode modes
- System mode
- Undefined mode
- User mode Non privileged mode

The processor enters

- Abort mode: when there is a failed attempt to access memory.
- Fast interrupt : Two interrupt levels
- Interrupt : available on the ARM processor
- Supervisor : The processor is in after reset and is generally the mode that an operating

system kernel operates in.

- System mode: A special version of user mode that allows full read-write access to the cpsr.
- Undefined mode : When the processor encounters an instruction that is undefined or not

supported by the implementation.

- User mode: The mode is used for programs and applications.

Jumper settings

Jumper	Description
Jp3	Led enable: place a jumper

J6	Adc selection: place a jumper for the required channel; this will connect lpc4088 pins with pot or lm35.
Jp7	Ispenable
Jp10	Connects the buzzer with p0.26 or rtc alarm output pin
Jp11	Rts and dtr pin termination, jumper not needed
Jp13	Ft232 enable: powers the ft232 ic

Config dip switch (sw29)

Dip switch pin	Pin	Description
5v	1-step	Power supply for stepper motor
5v	2-lcd	Power supply for lcd
5v	3-eep	Power supply for i2c eeprom
Zigbee	4-zbe	Power supply for zigbee
Ft232	5 and 6	Connects ft232with p0.0 and p0.1
Ft232	7 and 8	Connects ft232with p0.2 and p0.3 (default)

Power supply

Theexternalpowershouldbedc5v,1a.
Thearmboardproduces+3.3vusinganonboardvoltageregulator,whichprovidessupplytothe arm controller. Powersupply is controlled through slideswitchsw1. The 5v volt from usb or dc jack is used for

Peripherals directly

Power switch	Ext usb	Ext supply turned on
	Ext usb	Usb supply turned on

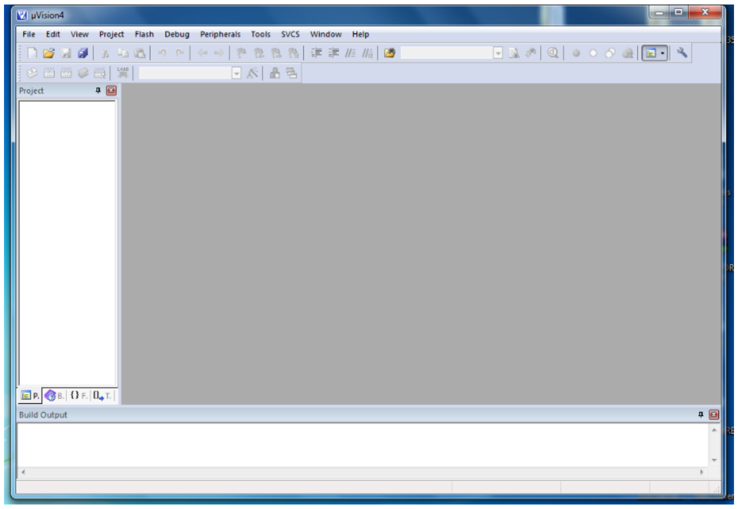
Power supply to the peripherals

Peripheral	Switch	Pin	Description
Lpc4088	Sw1	-	Turn on the switch sw1 towards usb or ext
Ft232	Jp13	-	Place a jumper at jp13
Stepper	Sw29	1	Turn on the dip switch pin

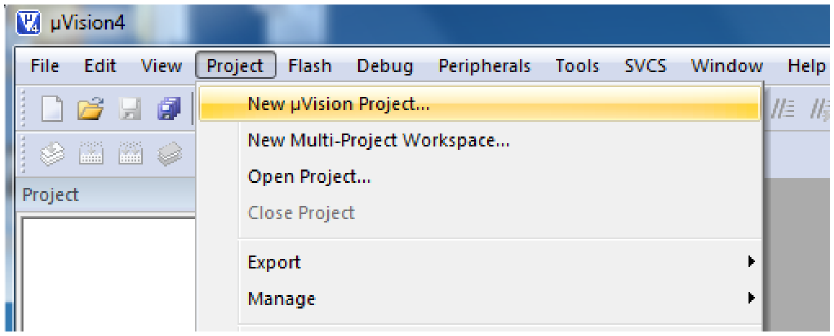
Lcd	Sw29	2	Turn on the dip switch pin
I2c eeprom	Sw29	3	Turn on the dip switch pin
Zigbee	Sw29	4	Turn on the dip switch pin

CREATING A PROJECT IN KEIL

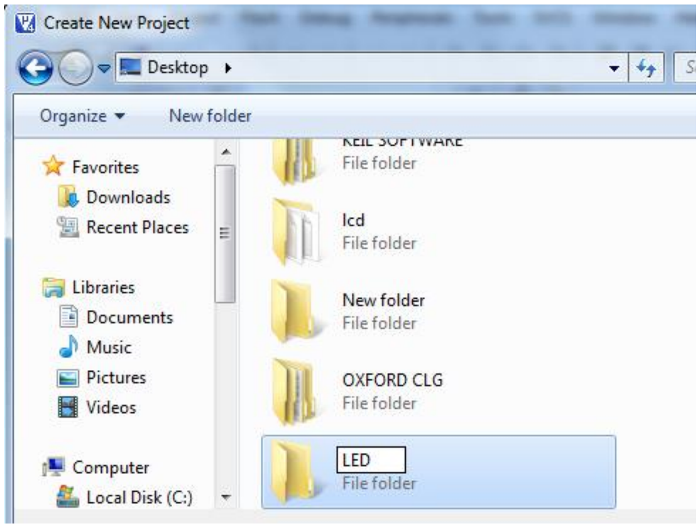
Step1: open keil and the environment will be as following


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Project
Build Output

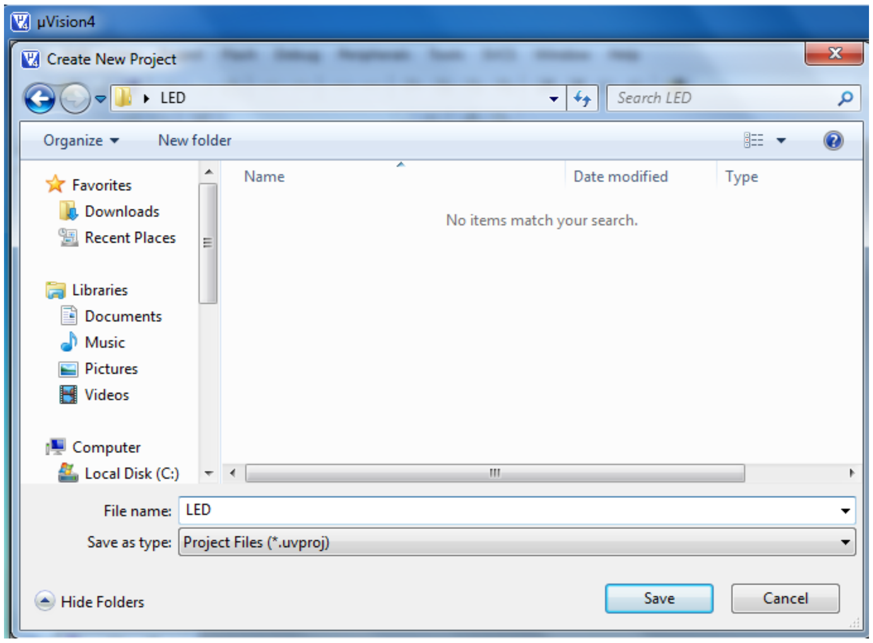
Step2: select project→new uvision project


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Close Project
Export
Manage
Project

Step3: create a project folder



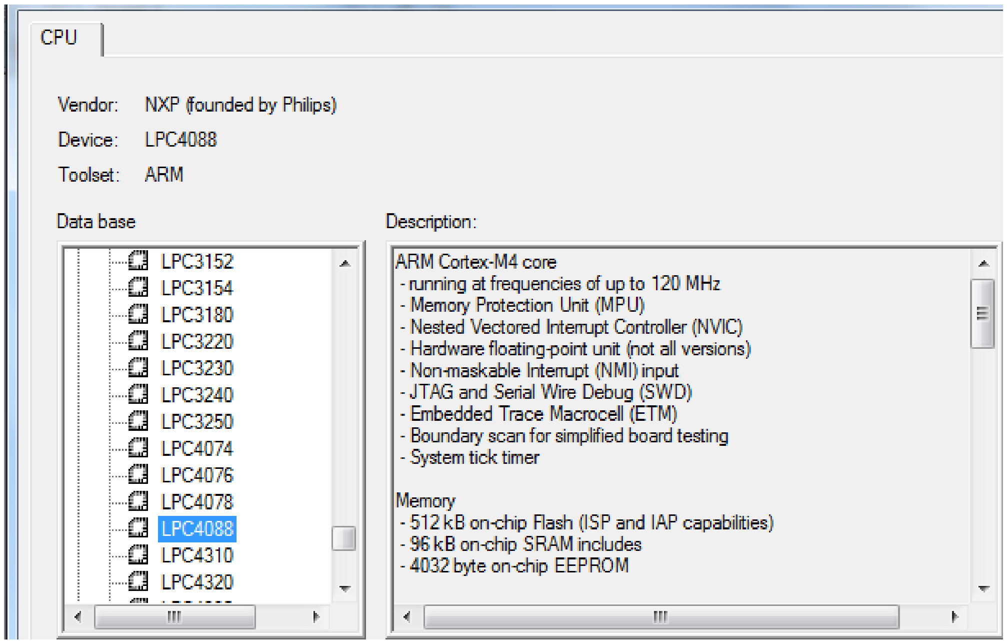
Step4: name the project



File name: LED

Save as type: Project Files (*.uvproj)

Step5: select the processor and click ok



CPU

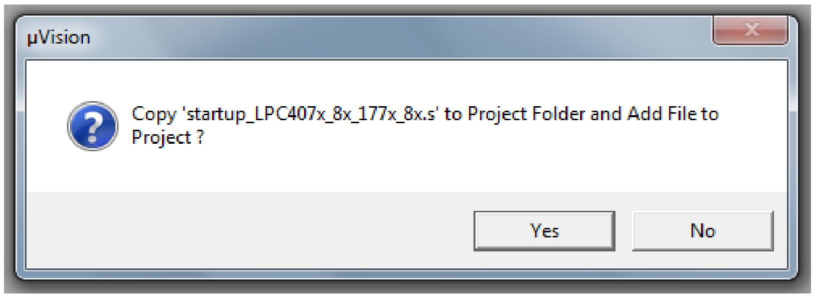
Vendor: NXP (founded by Philips)
Device: LPC4088
Toolset: ARM

Data base: LPC3152, LPC3154, LPC3180, LPC3220, LPC3230, LPC3240, LPC3250, LPC4074, LPC4076, LPC4078, LPC4088, LPC4310, LPC4320

Description:
ARM Cortex-M4 core
- running at frequencies of up to 120 MHz
- Memory Protection Unit (MPU)
- Nested Vectored Interrupt Controller (NVIC)
- Hardware floating-point unit (not all versions)
- Non-maskable Interrupt (NMI) input
- JTAG and Serial Wire Debug (SWD)
- Embedded Trace Macrocell (ETM)
- Boundary scan for simplified board testing
- System tick timer

Memory
- 512 kB on-chip Flash (ISP and IAP capabilities)
- 96 kB on-chip SRAM includes
- 4032 byte on-chip EEPROM

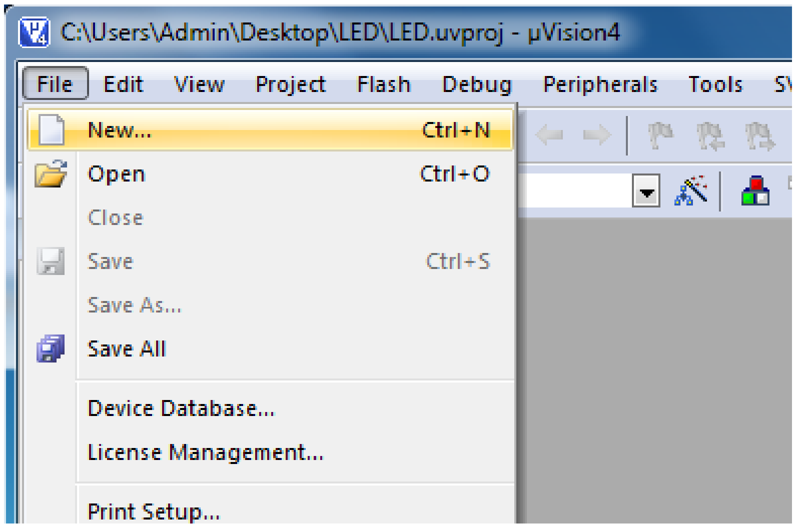
Step 6: click yes to the following message box



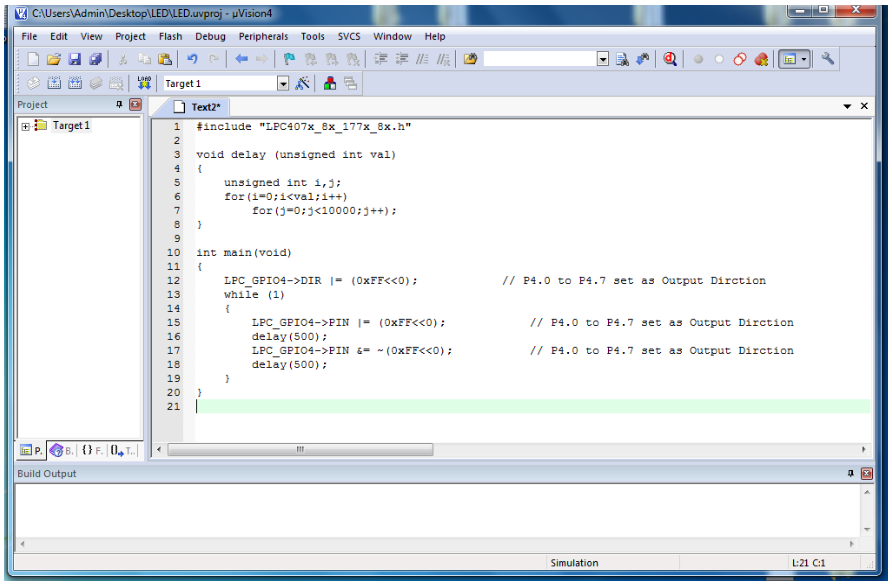
µVision

Copy 'startup_LPC407x_8x_177x_8x.s' to Project Folder and Add File to Project ?

Step 7: create a new source file by selecting file→new



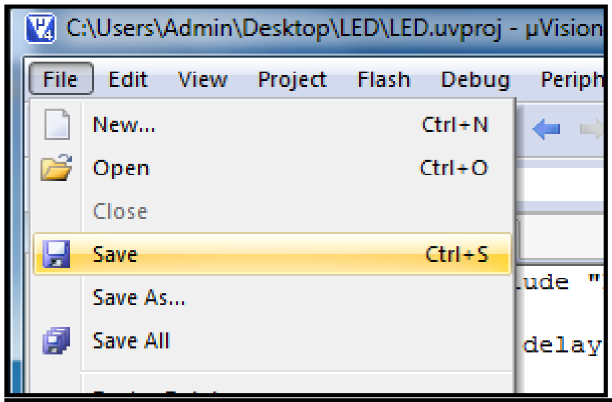
Step8: write the required source code



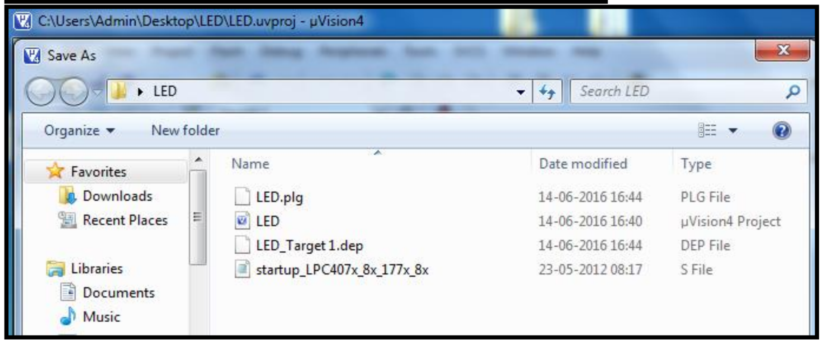
C:\Users\Admin\Desktop\LED\LED.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Target 1
Project
Target 1
Text2*
Build Output
Simulation
L:21 C:1

```c
#include "LPC407x_8x_177x_8x.h"

void delay (unsigned int val)
{
    unsigned int i,j;
    for(i=0;i<val;i++)
        for(j=0;j<10000;j++);
}

int main(void)
{
    LPC_GPIO4->DIR |= (0xFF<<0);            // P4.0 to P4.7 set as Output Dirction
    while (1)
    {
        LPC_GPIO4->PIN |= (0xFF<<0);            // P4.0 to P4.7 set as Output Dirction
        delay(500);
        LPC_GPIO4->PIN &= ~(0xFF<<0);           // P4.0 to P4.7 set as Output Dirction
        delay(500);
    }
}
```

Step 9: save this file with ".c" extension

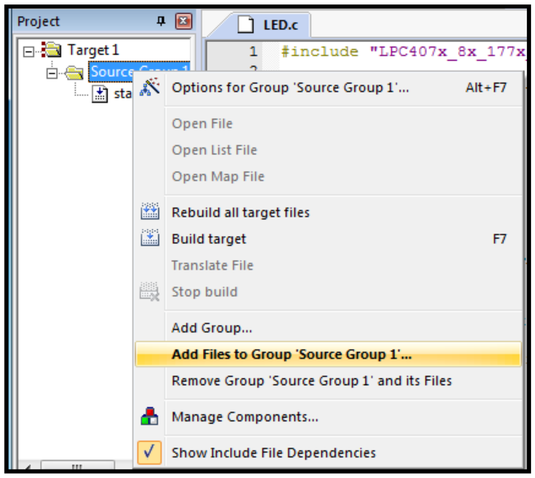


C:\Users\Admin\Desktop\LED\LED.uvproj - µVision
File Edit View Project Flash Debug Periph
New... Ctrl+N
Open Ctrl+O
Close
Save Ctrl+S
Save As...
Save All
ude "
delay



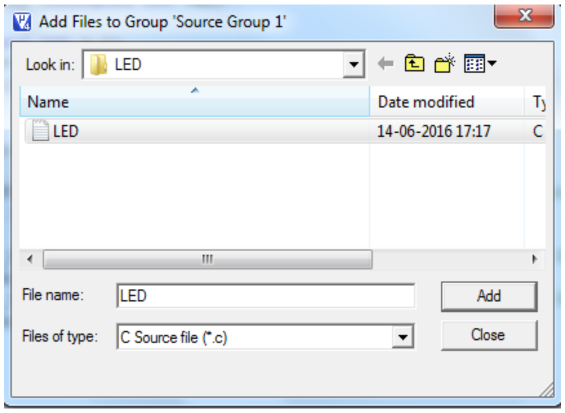
Name	Date modified	Type
LED.plg	14-06-2016 16:44	PLG File
LED	14-06-2016 16:40	µVision4 Project
LED_Target1.dep	14-06-2016 16:44	DEP File
startup_LPC407x_8x_177x_8x	23-05-2012 08:17	S File

Step 11: right click on source group and select add files to group



```
#include "LPC407x_8x_177x
```

Step 12: select the source code file, click add and then click close

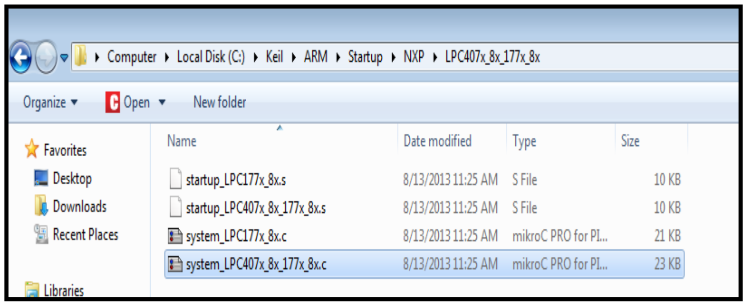


Name	Date modified	T
LED	14-06-2016 17:17	C

File name: LED

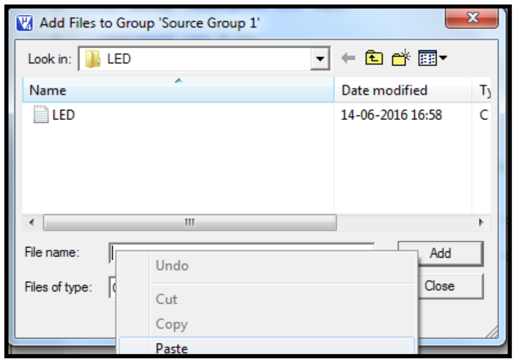
Files of type: C Source file (*.c)

Step 13: navigate to the folder "c:\keil\arm\startup\nxp\lpc407x_8x_177x_8x" and copy the file "system_lpc407x_8x_177x_8x.c"


Computer ▸ Local Disk (C:) ▸ Keil ▸ ARM ▸ Startup ▸ NXP ▸ LPC407x_8x_177x_8x
Organize
Open
New folder
Favorites
Desktop
Downloads
Recent Places
Libraries

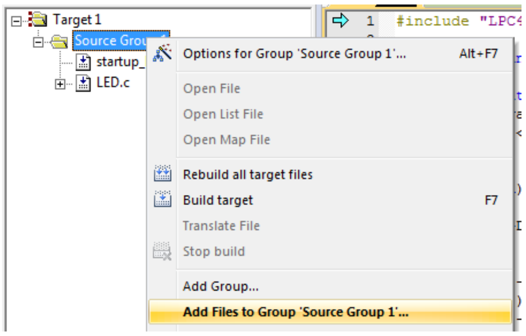
Name	Date modified	Type	Size
startup_LPC177x_8x.s	8/13/2013 11:25 AM	S File	10 KB
startup_LPC407x_8x_177x_8x.s	8/13/2013 11:25 AM	S File	10 KB
system_LPC177x_8x.c	8/13/2013 11:25 AM	mikroC PRO for PI...	21 KB
system_LPC407x_8x_177x_8x.c	8/13/2013 11:25 AM	mikroC PRO for PI...	23 KB

Step 14: paste it in your project directory

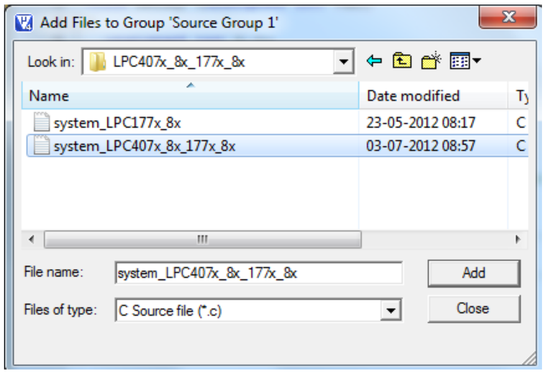


Name	Date modified	T
LED	14-06-2016 16:58	C

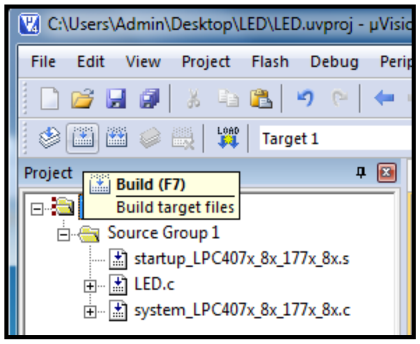
Step 15: right click on source group once again and select add files to group



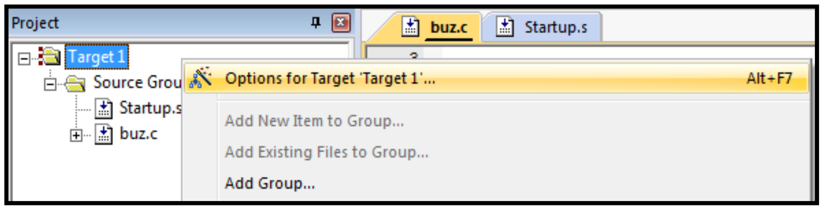
Step 16: now add this "system_lpc407x_8x_177x_8x.c" file to your project


Add Files to Group 'Source Group 1'
Look in: LPC407x_8x_177x_8x
Name
Date modified
system_LPC177x_8x 23-05-2012 08:17
system_LPC407x_8x_177x_8x 03-07-2012 08:57
File name: system_LPC407x_8x_177x_8x
Files of type: C Source file (*.c)
Add
Close

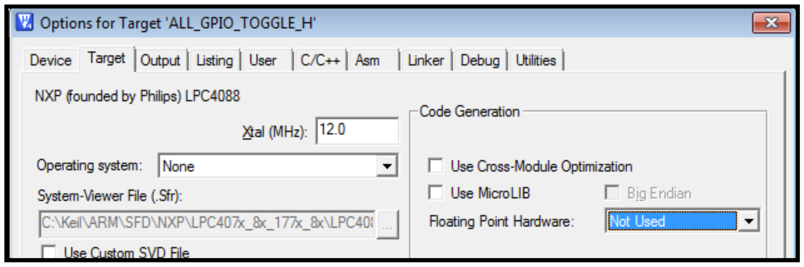
Step 17: build the project


C:\Users\Admin\Desktop\LED\LED.uvproj - µVisio
File Edit View Project Flash Debug Perip
Target 1
Project
Build (F7)
Build target files
Source Group 1
startup_LPC407x_8x_177x_8x.s
LED.c
system_LPC407x_8x_177x_8x.c

Step 18: right click "target1" on the project tab and select options for target



Step 19: on the target tab, select the floating point hardware as “not used”


Options for Target 'ALL_GPIO_TOGGLE_H'

NXP (founded by Philips) LPC4088

Xtal (MHz): 12.0

Operating system: None

System-Viewer File (.Sfr): C:\Keil\ARM\SFD\NXP\LPC407x_8x_177x_8x\LPC40

Use Custom SVD File

Code Generation

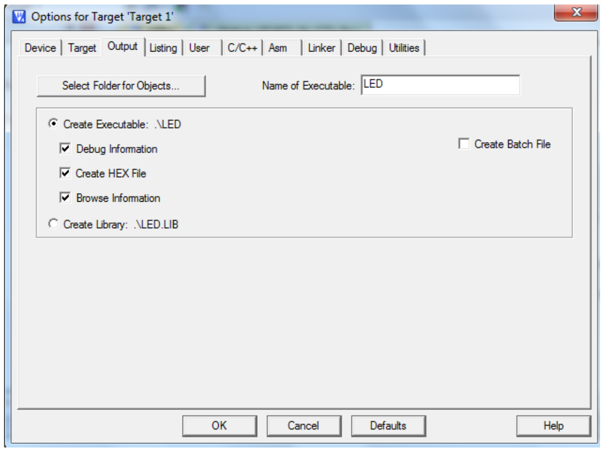
Use Cross-Module Optimization

Use MicroLIB

Big Endian

Floating Point Hardware: Not Used

Step 20: select output tab and put a check mark on “create hex file”. The name in the text box “name of executable” will be used to name the hex file.


Options for Target 'Target 1'

Name of Executable: LED

Create Executable: .\LED

☑ Debug Information

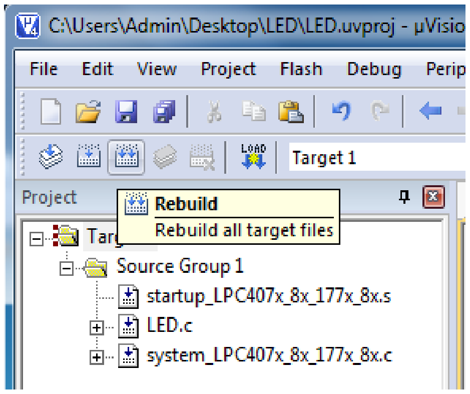
☑ Create HEX File

☑ Browse Information

Create Batch File

Create Library: .\LED.LIB

Step 20: click rebuild

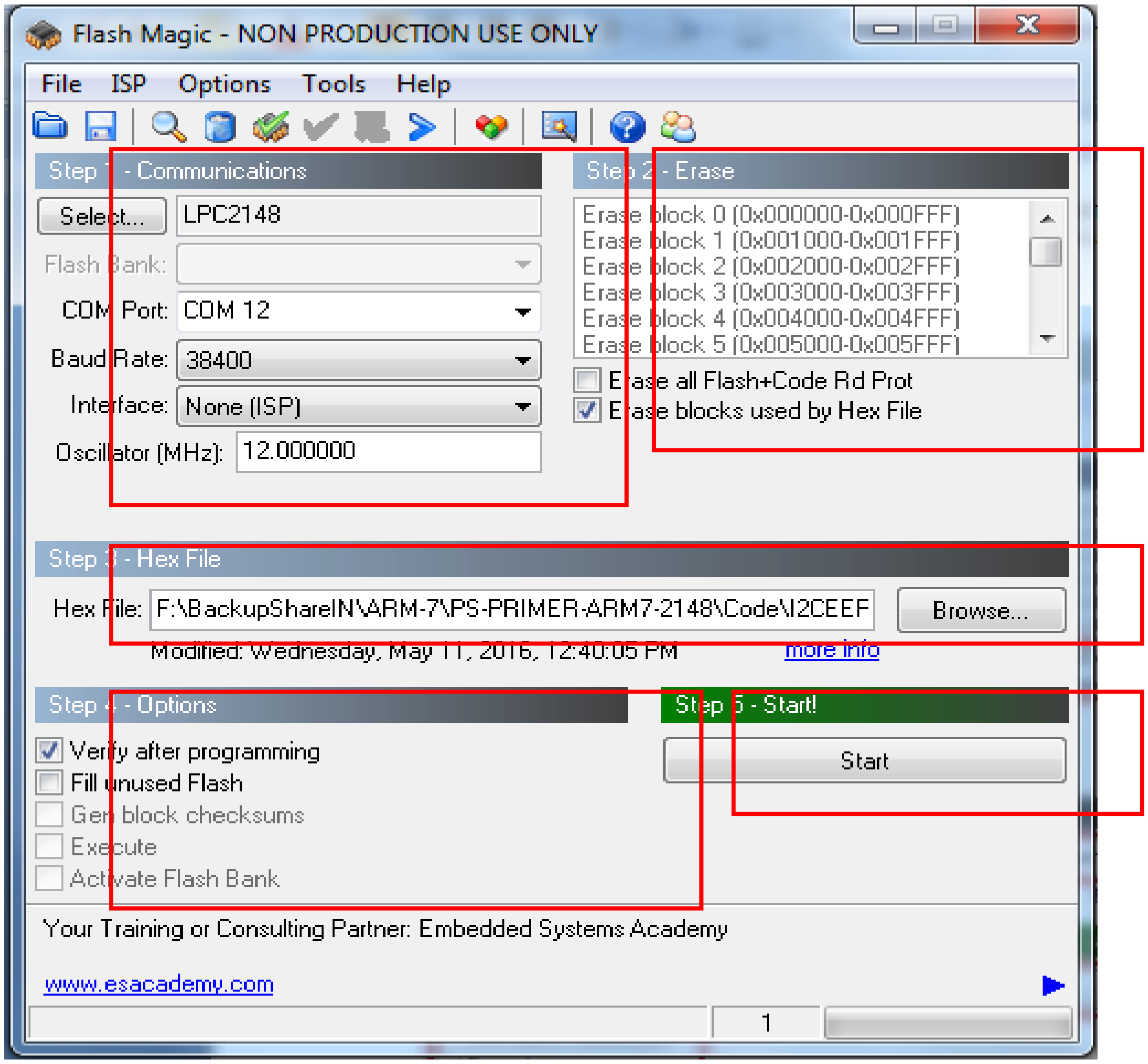


That’s all. Now write your hex file into arm cortex-m4 using flash magic

<u>PROGRAMMING IN FLASH MAGIC</u>

Hardware connection	Turn on dip switch sw29 pins 7 and 8. Supply power to the board using sw1. Press and hold sw4 (eint0) and press reset to enter into boot-loader mode

Five steps programming


Flash Magic - NON PRODUCTION USE ONLY

File ISP Options Tools Help

Step 1 - Communications

Select... LPC2148

Flash Bank:

COM Port: COM 12

Baud Rate: 38400

Interface: None (ISP)

Oscillator (MHz): 12.000000

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

☐ Erase all Flash+Code Rd Prot

☑ Erase blocks used by Hex File

Step 3 - Hex File

Hex File: F:\BackupShareIN\ARM-7\PS-PRIMER-ARM7-2148\Code\I2CEEF Browse...

Modified: Wednesday, May 11, 2016, 12:40:05 PM more info

Step 4 - Options

☑ Verify after programming

☐ Fill unused Flash

☐ Gen block checksums

☐ Execute

☐ Activate Flash Bank

Step 5 - Start!

Start

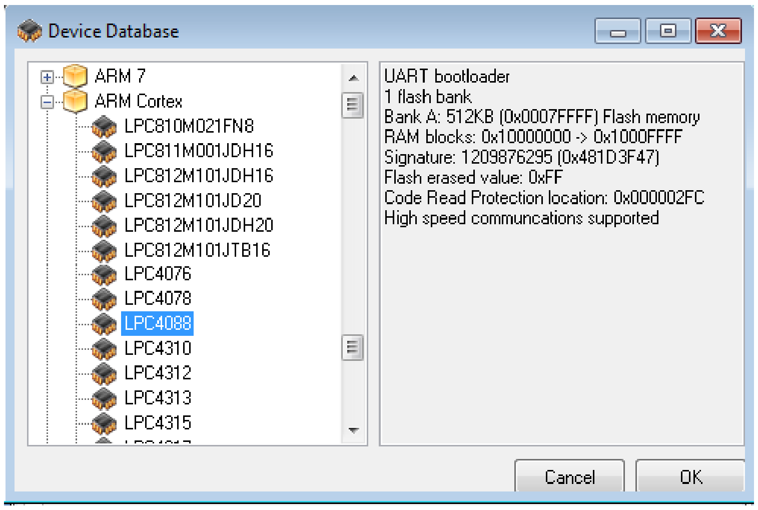
Your Training or Consulting Partner: Embedded Systems Academy

www.esacademy.com

1

Step 1 – communications:

1. Click select…,
2. Expand arm cortex from device database
3. Scroll down and select your ic.
4. Click ok.

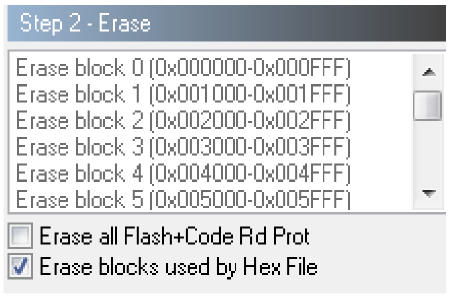

Device Database

ARM 7
ARM Cortex
LPC810M021FN8
LPC811M001JDH16
LPC812M101JDH16
LPC812M101JD20
LPC812M101JDH20
LPC812M101JTB16
LPC4076
LPC4078
LPC4088
LPC4310
LPC4312
LPC4313
LPC4315

UART bootloader
1 flash bank
Bank A: 512KB (0x0007FFFF) Flash memory
RAM blocks: 0x10000000 -> 0x1000FFFF
Signature: 1209876295 (0x481D3F47)
Flash erased value: 0xFF
Code Read Protection location: 0x000002FC
High speed communcations supported

Step 2 – erase

Put a check mark on erase blocks used by hex file checkbox

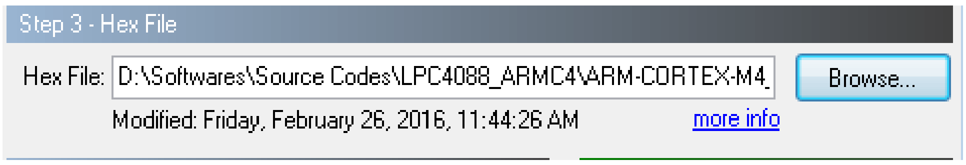

Step 2 - Erase

Erase block 0 (0x000000-0x000FFF)
Erase block 1 (0x001000-0x001FFF)
Erase block 2 (0x002000-0x002FFF)
Erase block 3 (0x003000-0x003FFF)
Erase block 4 (0x004000-0x004FFF)
Erase block 5 (0x005000-0x005FFF)

☐ Erase all Flash+Code Rd Prot
☑ Erase blocks used by Hex File

Step 3 – hex file

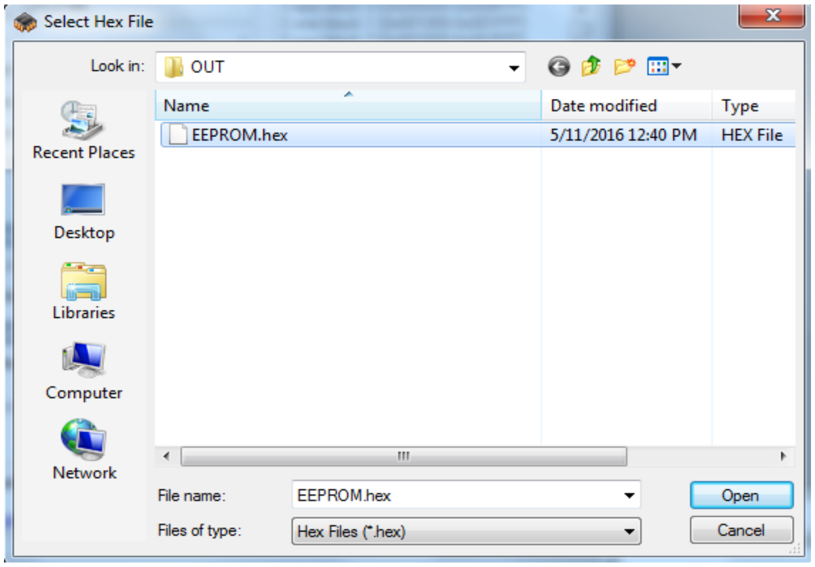
1. Click browse…


Step 3 - Hex File

Hex File: D:\Softwares\Source Codes\LPC4088_ARMC4\ARM-CORTEX-M4_

Modified: Friday, February 26, 2016, 11:44:26 AM

2. Navigate to the project folder; select the hex file to be loaded and click open.


Select Hex File

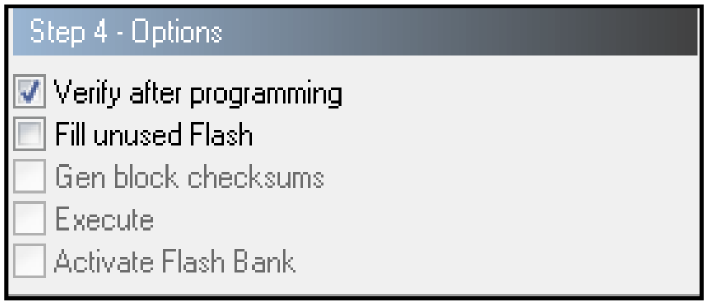
Look in: OUT

Name	Date modified	Type
EEPROM.hex	5/11/2016 12:40 PM	HEX File

File name: EEPROM.hex
Files of type: Hex Files (*.hex)

Step 4 – options

1. Put a check mark on each options that is required for the project



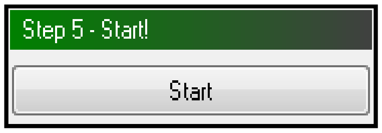
Step 4 - Options

- [✓] Verify after programming
- [ ] Fill unused Flash
- [ ] Gen block checksums
- [ ] Execute
- [ ] Activate Flash Bank

Generally verify after programming is required to cross check the burned hex file with the loaded hex file and the remaining options left unchecked.

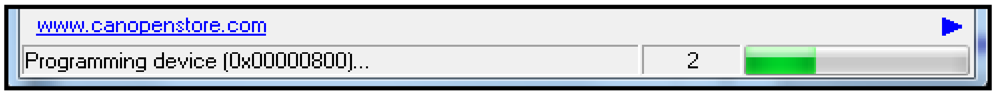
Step 5 – start

1. Click start. This will start programming the chip



2. See the status bar for current status

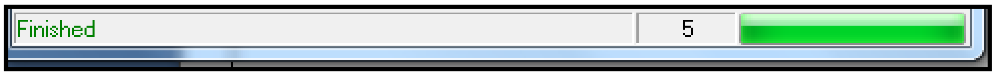
Programming:



www.canopenstore.com

Programming device (0x00000800)... 2

Programming finished:



Finished 5

Execution

Press reset after programming

Pre-requisites:

Place a Jumper at JP13 and Turn ON the DIP switch SW29 pins 7 and 8.

LAB Viva QUESTIONS:

1. What is an embedded system?
2. Mention the difference between microprocessor and microcontroller.
3. Enumerate the terms object oriented and object based language
4. Define Pipelining.
5. List the basic units of Microcontroller
6. What is the difference between embedded systems and the system in which rtos is running?
7. Discuss about semaphore.
8. What are the instructions used to access the memory in ARM?
9. Mention the characteristics of RISK Instruction.
10. Define Interrupt.

Result:

Thus the study of ARM Evaluation is completed.

EX.NO: 3 FLASHING OF LEDS USING ARM PROCESSOR

To perform an experiment on Flashing of LEDby interfacing with ARM processor..

COMPONENTS REQUIRED:

ARM Trainer Kit, Mini USB cable

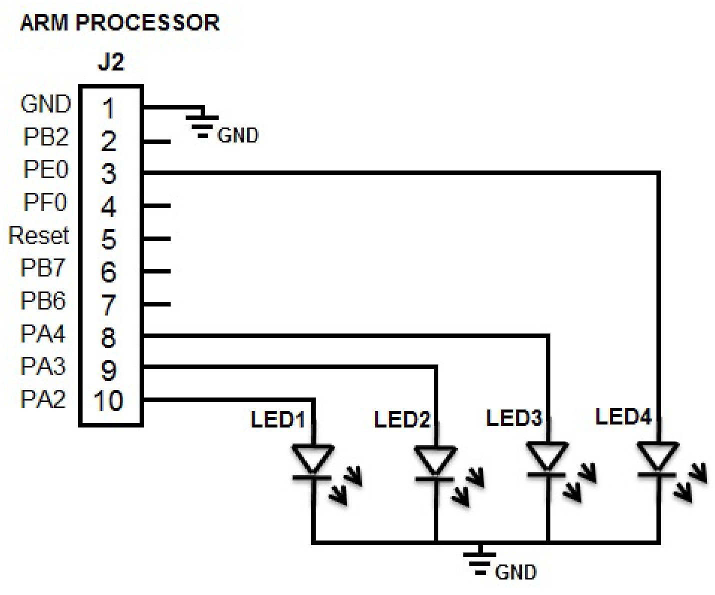
THEORY:

5 LEDs are connected with ARM processor by individual digital pin. rogram is written to ON LEDs one by one then off all the LEDs at a time, then ON the EDs one by one, the same loop is operated. So the flashing of LEDs is done.

PROCEDURE:

1. Open Keil Software in PC and create a project.
2. Select the processor – ARM LPC2148 and click ok
3. Create a new source file
4. Write the required source code and save.
5. Add files to source group
6. Build the project
7. Debug the project, and select peripheral GPIO Slow Interface
8. Compile and run the project
9. Create Hex file and rebuild the target.
10. Now, Connect LPC2148 kit via USB port
11. Open Flash Magic to interface with LPC2148.
12. Using Flash magic Download the hex file and start execution
13. Once executed successfully, Check for output in the kit.

CIRCUIT DIAGRAM:



Program Coding (Sheet attached)

VIVA QUESTIONS:

1. What is seven segment displays?
2. Where LEDs are used?
3. What are the different configurations of LED?
4. What is the use of flash magic software?
5. Differentiate LED from LCD.

Sample Output

RESULT:

Thus the flashing of LEDs using ARM processor is performed and verified.

EX.NO: 4 INTERFACING DIGITAL TO ANALOG CONVERTER WITH ARM PROCESSOR

AIM:

To convert digital signal to analog signal by interfacing with ARM processor.

COMPONENTS REQUIRED:

ARM Trainer Kit, Mini USB cable

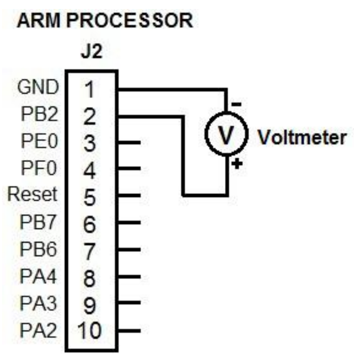
THEORY:

Digital scale value in program is uploaded to ARM processor. This value is passed to digital to analog converting pin using analog write program line, so equivalent digital value is obtained at the output of that pin. This is monitored by Voltmeter connected across the same pin with ground.

PROCEDURE:

1. Open Keil Software in PC and create a project.
2. Select the processor – ARM LPC2148 and click ok
3. Create a new source file
4. Write the required source code and save.
5. Add files to source group
6. Build the project
7. Debug the project, and select peripheral GPIO Slow Interface
8. Compile and run the project
9. Create Hex file and rebuild the target.
10. Now, Connect LPC2148 kit via USB port
11. Open Flash Magic to interface with LPC2148.
12. Using Flash magic Download the hex file and start execution
13. Once executed successfully, Check for output in the kit.

CIRCUIT DIAGRAM:



Program Coding (Sheet attached)

LAB VIVA QUESTIONS:

1. List the types of ADC and DAC
2. Define resolution.
3. Summarize the features of Conversion time in ADC.
4. What is the function of Sample-and-hold circuits in analog-to digital converters?
5. Why are internal ADCs preferred over external ADCs?
6. What are the ADC operating modes in LPC2148?
7. What is the function of A/D Status Register?
8. Which pin provides a voltage reference level for the D/A converter?
9. What is Burst conversion mode?
10. What is settling time?

Sample Output

RESULT:

Thus the Conversion of Digital to Analog is performed and verified with ARM processor successfully

Ex.No: 5 INTERFACE LED AND PWM AND TO VERIFY THE OUTPUT IN THE ARM7

AIM:

To switch ON and OFF the LED by interfacing with the ARM processor.

COMPONENTS REQUIRED:

ARM trainer Kit, Mini USB cable

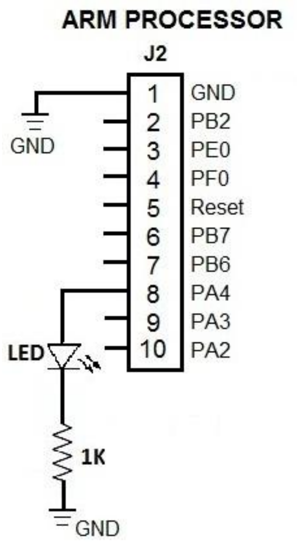
THEORY:

ARM processor is used for LED ON and OFF. Digital out of processor is connected to the LED. When digital out is HIGH, LED gets ON, When digital out is LOW, LED gets OFF. Delay is used in the program in between LED on and off to view the led change of state from ON to OFF.

PROCEDURE:

1. Open Keil Software in PC and create a project.
2. Select the processor – ARM LPC2148 and click ok
3. Create a new source file
4. Write the required source code and save.
5. Add files to source group
6. Build the project
7. Debug the project, and select peripheral GPIO Slow Interface
8. Compile and run the project
9. Create Hex file and rebuild the target.
10. Now, Connect LPC2148 kit via USB port
11. Open Flash Magic to interface with LPC2148.
12. Using Flash magic Download the hex file and start execution
13. Once executed successfully, Check for output in the kit.

CIRCUIT DIAGRAM:



Program Coding (Sheet attached)

Sample Output

RESULT:
Thus the interfacing of LED with ARM processor is done successfully

EX.NO : 6 INTERFACING REAL TIME CLOCK WITH ARM PROCESSOR

AIM:

To perform the experiment to study the use the internal RTC module by interfacing with ARM processor.

COMPONENTS REQUIRED:

ARM Trainer Kit , Mini USB cable, LCD

THEORY:

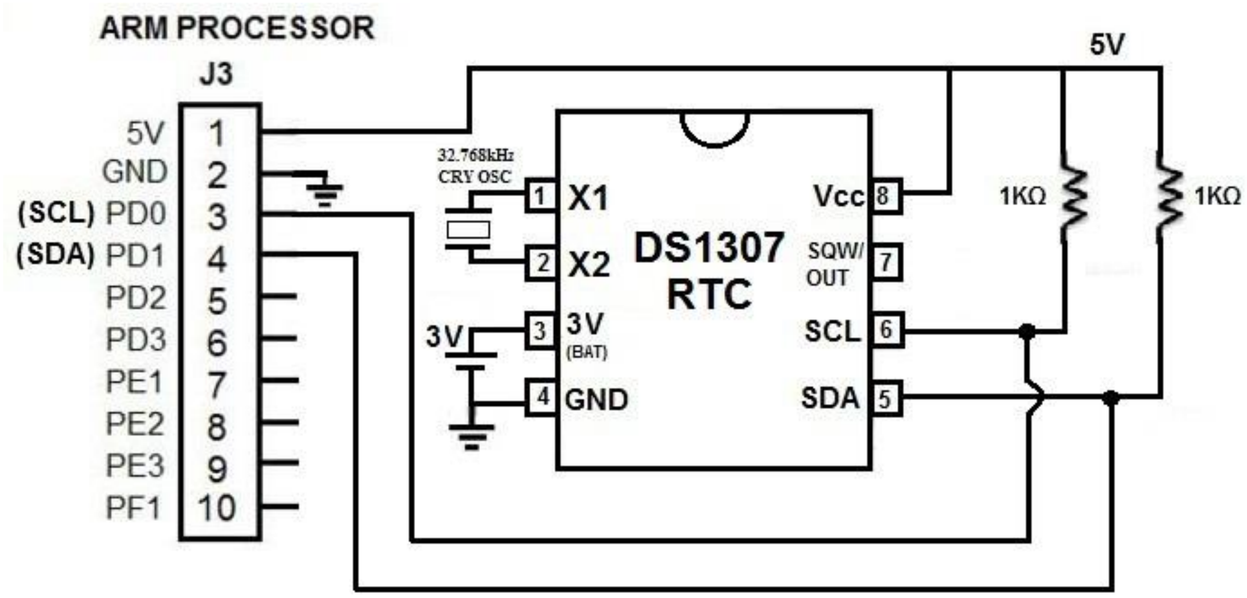
Real-time clock (RTC) counts seconds, minutes, hours, day of the week, date of the month, month, and year with leap-year. Full binary-coded decimal (BCD) clock/calendar and 56-byte, battery-backed, nonvolatile (NV) RAM for data storage. Two-wire serial interface are SDA, SCL. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

PROCEDURE:

1. Power: In the PCONP register, set bit PCRTC.
2. Select the Clock from RTC oscillator.
3. Initialize RTC and set all their register to zero
4. Enable RTC
5. Set current time and date
6. Set alarm time.
7. Enable RTC interrupt
8. Get time and date and display its value in LCD

- Open Keil Software in PC and create a project.
- Select the processor – ARM LPC2148 and click ok
- Create a new source file
- Write the required source code and save.
- Add files to source group
- Build the project
- Debug the project, and select peripheral GPIO Slow Interface
- Compile and run the project
- Create Hex file and rebuild the target.
- Now, Connect LPC2148 kit via USB port
- Open Flash Magic to interface with LPC2148.
- Using Flash magic Download the hex file and start execution
- Once executed successfully, Check for output in the kit

CIRCUIT DIAGRAM:



Program Coding (Sheet attached)

Sample Output

RESULT:

Thus the interfacing of Real Time Clock with ARM processor is performed

Ex.No : 7 INTERFACING KEYBOARD AND LCD

AIM:

To perform experiment on 4x4 matrix keypad and display its result in LCD by interfacing with ARM processor.

COMPONENTS REQUIRED:

PS- CORTEX-M4-TYRO-V4r2, Mini USB cable, LCD

THEORY:

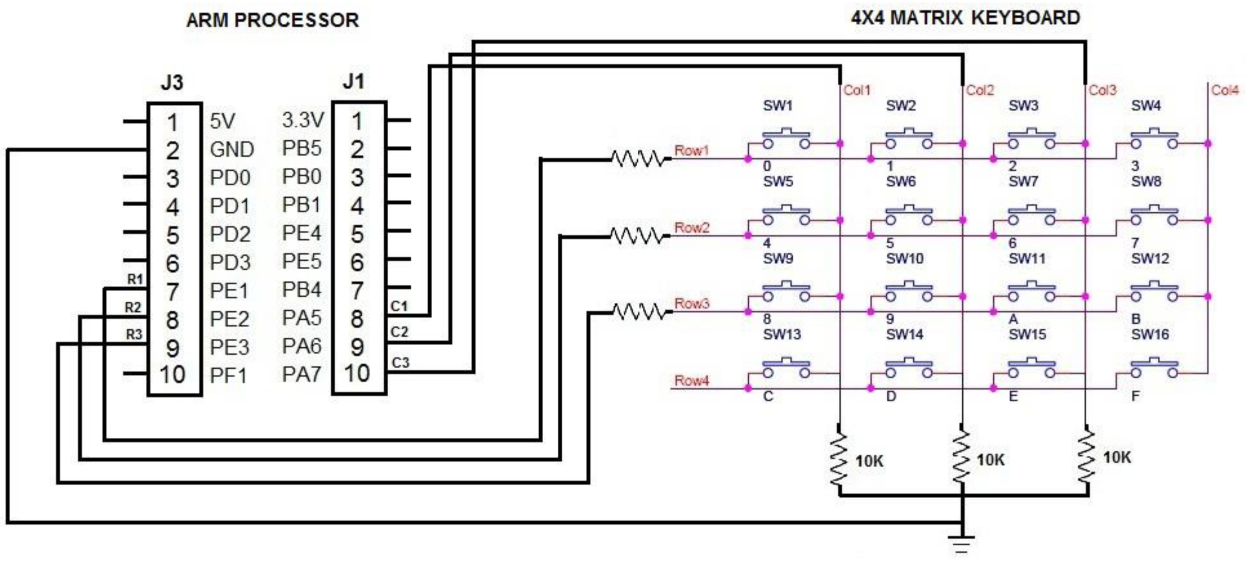
Keyboard is scanned and the pressed key is detected then it is printed. When the columns pins of ARM processor is set in output mode, 5v is supplied by default to this pins. 10k resistor is connected in each column and another end of all this resistor is connected to ground.To start scanning, Column1 is set as high, remaining columns set as low, so 5v is available in column1 alone. When scanning all the rows, if any key is pressed in row1 will received 5v in row1, so column1 row1 key is the pressed key, it is assumed as value of this position is 1, so 1 is printed. If row2 key is pressed, 5v in column1 is received in row2, so the pressed key is column1 row2, it is assumed as 4 as it is the 3x3 matrix, this process is continued to row3. Next step column2 is enabled by setting it as high and all the rows are scanned and it key values areassumed like 2,5,8 if it is pressed.

PROCEDURE:

1. Power: In the PCONP register, set the PCGPIO bit
2. Configure all the row pins as output and make it HIGH
3. Configure all the column pins as input and make it HIGH
4. Configure all the LCD pins as output and write necessary software routines for LCD initialization, command and data
5. Make a row pin Zero and check all the four column lines for zero
6. If any key is pressed, the corresponding column will get zero
7. Read the key value and display it in LCD

   - Open Keil Software in PC and create a project.
   - Select the processor – ARM LPC2148 and click ok
   - Create a new source file
   - Write the required source code and save.
   - Add files to source group
   - Build the project
   - Debug the project, and select peripheral GPIO Slow Interface
   - Compile and run the project
   - Create Hex file and rebuild the target.
   - Now, Connect LPC2148 kit via USB port
   - Open Flash Magic to interface with LPC2148.
   - Using Flash magic Download the hex file and start execution
   - Once executed successfully, Check for output in the kit

CIRCUIT DIAGRAM:



PROGRAM:

Sample Output

RESULT:

Thus the keyboard interfacing with ARM processor is done and pressed key is verified successfully.

INTERFACING LCD WITH ARM PROCESSOR

AIM:

To display a string in LCD using 4 bit mode by interfacing with ARM processor.

COMPONENTS REQUIRED:

ARM Trainer Kit, Mini USB cable, LCD

THEORY:

VSS – GND
VDD – 5V (separate supply not from ARM processor)
V0 –3 wires 10kΩ pot, top wire to 5v, bottom wire to ground, middle wire to V0
RS – to arm processor digital pin
RW – to GND
E – to arm processor digital pin
D4,D5,D6,D7 – to arm processor digital pin
To glow up internal light of LCD
A – Anode to 5V supply
K – Cathode to GND
V0 – connected to 10kΩ pot, this is used to change brightness of LCD display.
ARM processor send RS-Reset and E-Enable signal to LCD display.
Then it sends the 4 bit digital data to LCD, to display text.

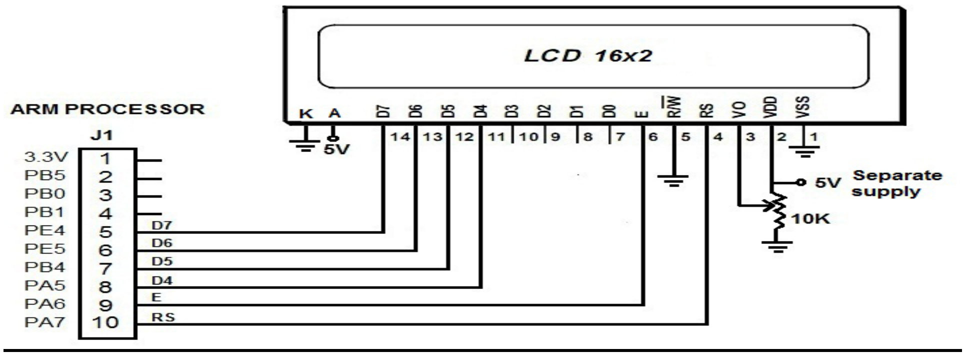
PROCEDURE:

1. Power: In the PCONP register, set the PCGPIO bit
2. Configure all the LCD pins as outputs
3. Store all the initialization command and data to be displayed in an array
4. Make RS pin low and send a command, pulse the EN pin ON/OFF for a moment
5. Repeat step 4 for all the initialization commands
6. Send First line command using step 4
7. Make RS pin high and send a character to be displayed, pulse the EN pin ON/OFF for a moment
8. Repeat step 5 until the end of first line (16 characters)
9. Send the second line command using step 4
10. Make RS pin high and send a character to be displayed, pulse the EN pin ON/OFF for a moment
11. Repeat step 5 until the end of second line (16 characters)

- Open Keil Software in PC and create a project.
- Select the processor – ARM LPC2148 and click ok
- Create a new source file
- Write the required source code and save.
- Add files to source group
- Build the project
- Debug the project, and select peripheral GPIO Slow Interface
- Compile and run the project
- Create Hex file and rebuild the target.
- Now, Connect LPC2148 kit via USB port
- Open Flash Magic to interface with LPC2148.

- Using Flash magic Download the hex file and start execution
- Once executed successfully, Check for output in the kit

CIRCUIT DIAGRAM:



PROGRAM:

LAB QUESTIONS:

1. Mention the function of pull up resistor?

2. Outline the keyboard matrix.
3. Summarize the working principal of LCD.
4. What kind of interrupt is generated if a key has to be operated in an interrupt mode?
5. How many rows and columns are present in a 16 x 2 alphanumeric LCD?

Sample Output

RESULT:

Thus the interfacing LCD display with ARM processor is done and text is displayed in LCD.

EX. NO : 8 INTERRUPT PERFORMANCE CHARACTERISTICS OF ARM AND FPGA

AIM:

To compare the performance of FPGA and CORTEX-M4 using interrupts.

COMPONENTS REQUIRED:

ARM Trainer Kit, 2 nos. of Mini USB cable, one FPGA stick board

THEORY:

Two interrupt ports are available in ARM processor. Two interrupt ports are PUSH1, PUSH2. PUSH1 is linked with PF4 and PUSH2 is linked with PF0, so external switch is to be connected with PF4 to trigger the PUSH1 interrupt, and PF0 for PUSH2.SW1 and SW2 switch are present in the ARM target board, SW1 can be used instead of connecting external switch to PF4 to trigger PUSH1 interrupt. SW2 can be used instead of connecting external switch to PF0 to trigger PUSH2 interrupt. These interrupts are initiated on RISING or FALLING of signal which is to provided through switches and as mentioned in program in "attach Interrupt" function.

PROCEDURE:

1. Configure the External Interrupt of CORTEX-4
2. Configure pins P4.0 to P4.7 as GPIO pins using IOCON registers IOCON_P4_0 to IOCON_P4_7
3. Configure the pins P4.0 to P4.7 as output pins using their direction registers
4. Configure Interrupt in FPGA
5. Apply a Finite no of pulses at the interrupt pins of CORTEX-M4 and FPGA
6. Capture the interrupt responses in both ARM and FPGA
7. If both processors output are same, then increase the frequency and compare the performance
8. Repeat until one of the processor fails to capture the interrupt signal

LAB QUESTIONS:

1. Define interrupts.
2. What is FPGA?
3. Difference between ARM and FPGA.
4. What are PROS and CONS for ARM?
5. What is interrupt pipelining?

RESULT:

Thus the implementation of interrupt performance of ARM and FPGA is performed.

EX.NO:9 INTERFACING STEPPER MOTOR WITH ARM PROCESSOR

AIM:

To run a stepper motor by interfacing with ARM processor.

COMPONENTS REQUIRED:

ARM Trainer Kit, Mini USB cable, Stepper motor

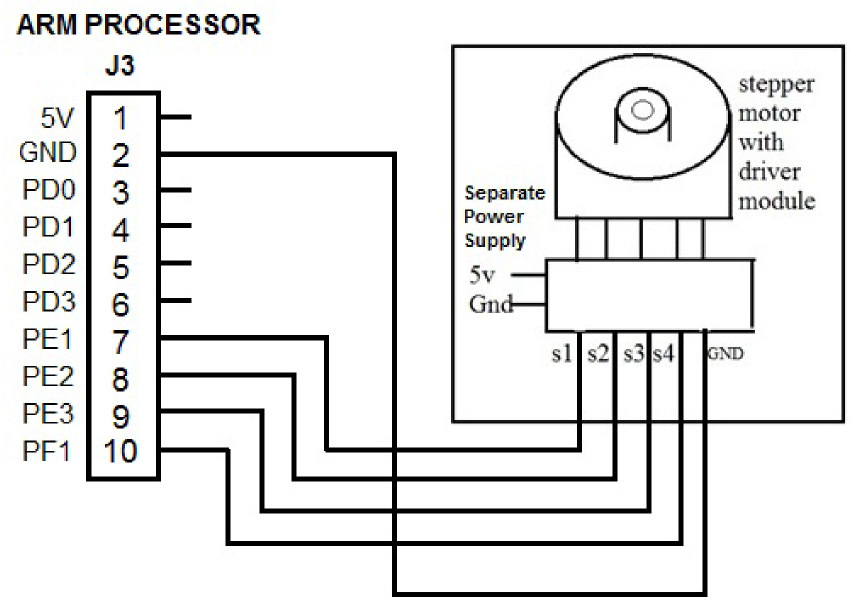
PROCEDURE:

1. Power: In the PCONP register, set the PCGPIO bit
2. Configure pins P3.23 to P3.26 as GPIO pins using IOCON registers IOCON_P3_23 to IOCON_P3_26
3. Configure the pins P3.23 to P3.24 as output pins using their direction registers.
4. Send the stepper sequence via IO lines
    - Open Keil Software in PC and create a project.
    - Select the processor – ARM LPC2148 and click ok
    - Create a new source file
    - Write the required source code and save.
    - Add files to source group
    - Build the project
    - Debug the project, and select peripheral GPIO Slow Interface
    - Compile and run the project
    - Create Hex file and rebuild the target.
    - Now, Connect LPC2148 kit via USB port
    - Open Flash Magic to interface with LPC2148.
    - Using Flash magic Download the hex file and start execution
    - Once executed successfully, Check for output in the kit

THEORY:

A stepper motor is a brushless, synchronous electric motor that converts digital pulses into mechanical shaft rotation. Every revolution of the stepper motor is divided into a discrete number of steps, and the motor must be sent a separate pulse for each step. Sequence 1100, 0110, 0011, 1001 sent to stepper from arm processor to rotate in forward direction, sequence is reversed to rotate motor in reverse direction.

CIRCUIT DIAGRAM:



VIVA QUESTIONS:

1. What is stepper motor?
2. Define step angle.
3. Mention applications of stepper motor.
4. What are the advantages and disadvantages of stepper motor?
5. Define holding torque.

Sample Output

RESULT:

Thus the stepper motor speed control is performed by interfacing stepper with ARM processor.

Ex.No : 10 TURN AN LED ON AND OFF WITH ARDUINO

AIM:

To Turn an LED light on and off with Arduino UNO

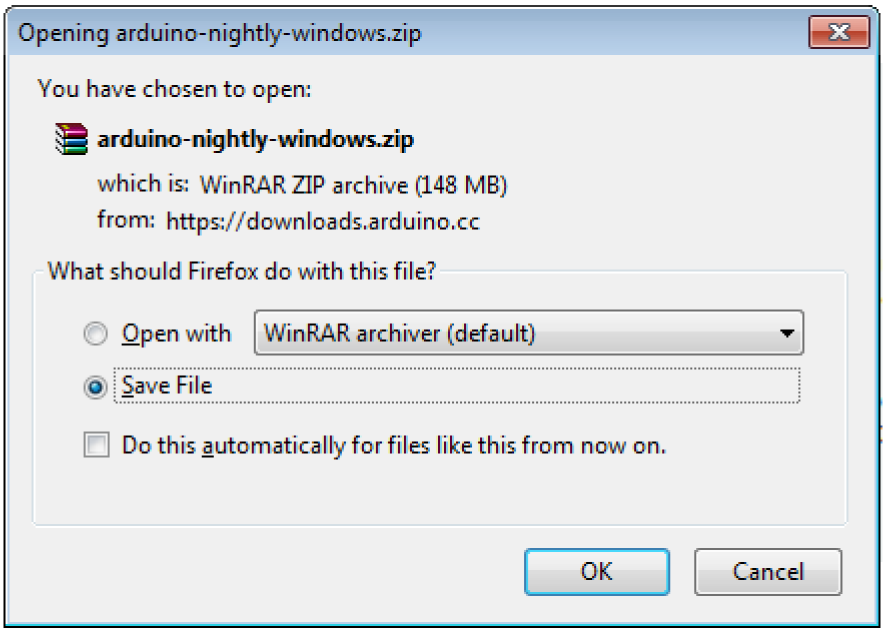
PROCEDURE:

1. Connect LED in the breadboard.
2. Connect GND to GND.
3. Connect DIGITAL PIN 7 to POSITIVE CONNECTION.
4. Now type the code in ARDUINO IDE and verify it.
5. Connect the USB of ARDUINO to the system.
6. Now upload the code in ARDUINO UNO board.
7. Check whether the LED is blinking or not.

ARDUINO INSTALLATION STEPS:

1. First you must have your Arduino board (you can choose your favorite board) and a USB cable. In case you use Arduino UNO, Arduino Duemilanove, Nano, Arduino Mega 2560, or Diecimila, you will need a standard USB cable (A plug to B plug), the kind you would connect to a USB printer as shown in the following image.

2. **Download Arduino IDE Software.**



Opening arduino-nightly-windows.zip

You have chosen to open:

arduino-nightly-windows.zip

which is: WinRAR ZIP archive (148 MB)

from: https://downloads.arduino.cc

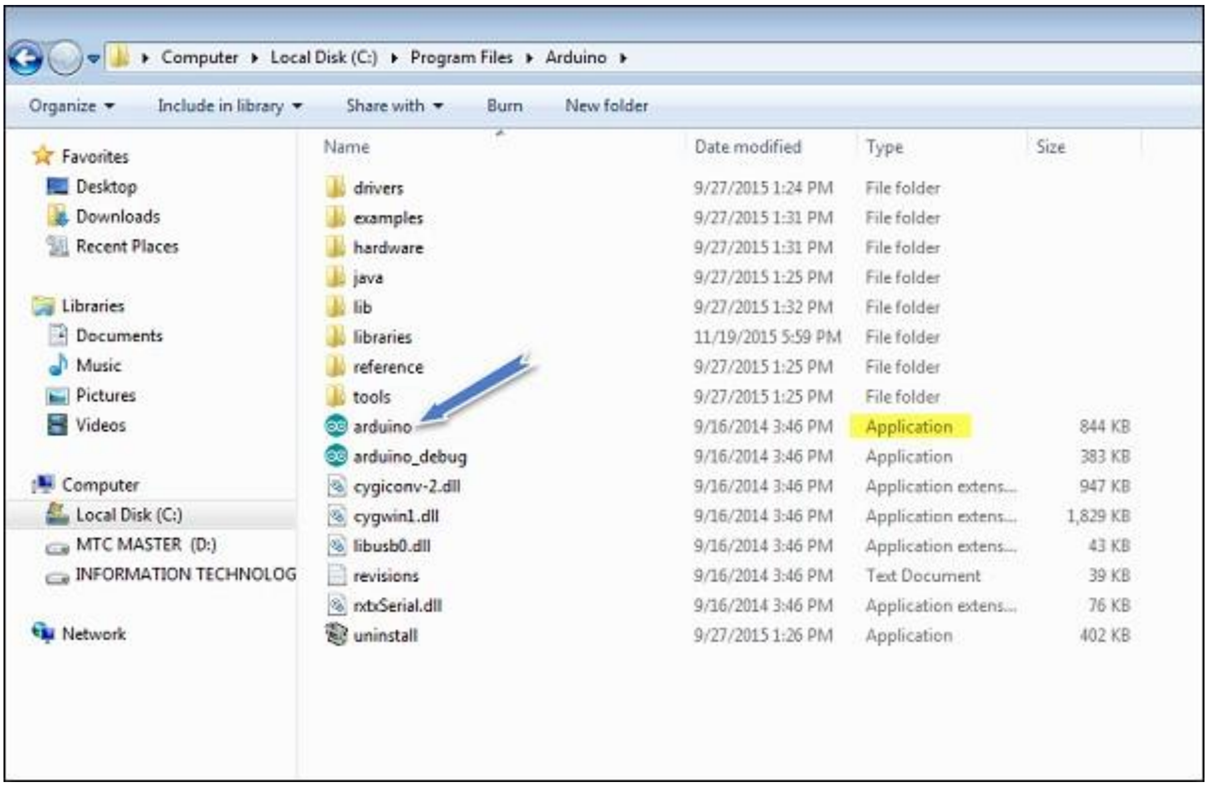
What should Firefox do with this file?

○ Open with [WinRAR archiver (default)]

● Save File

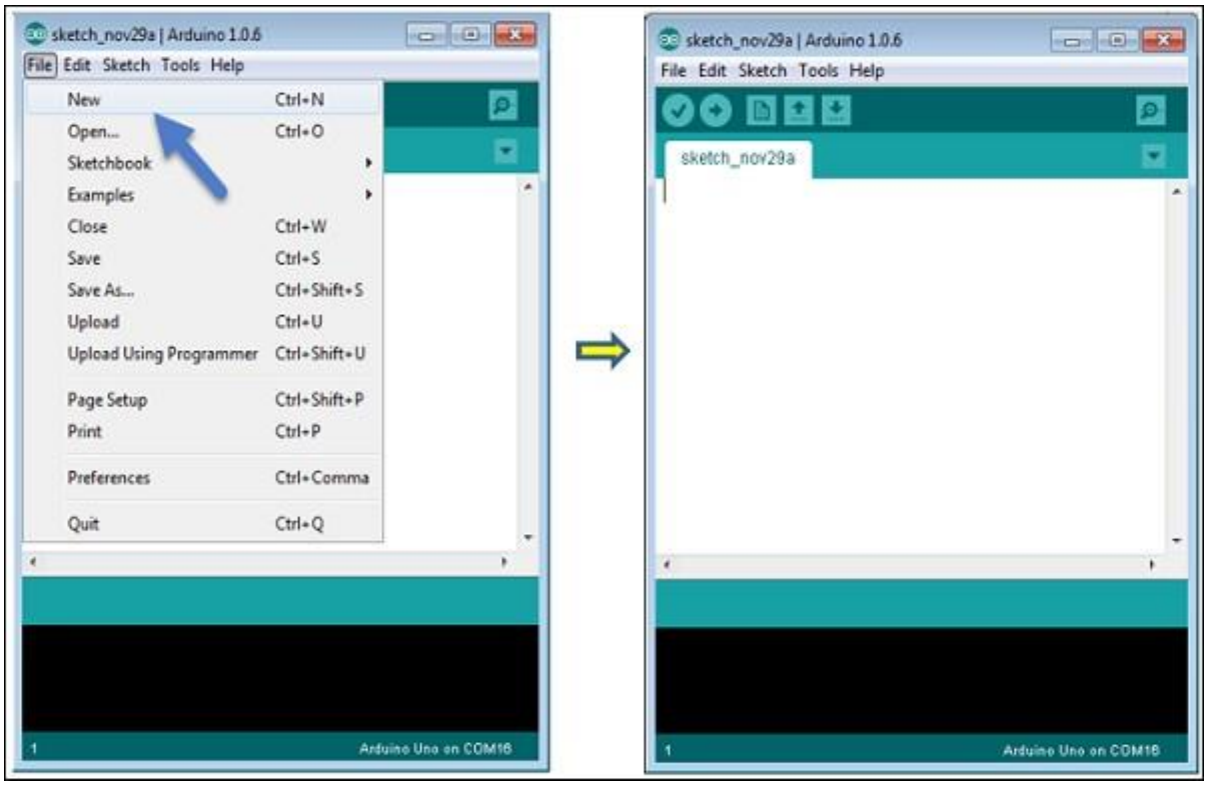
☐ Do this automatically for files like this from now on.

3. **Power up your board.**
4. **Launch Arduino IDE.**

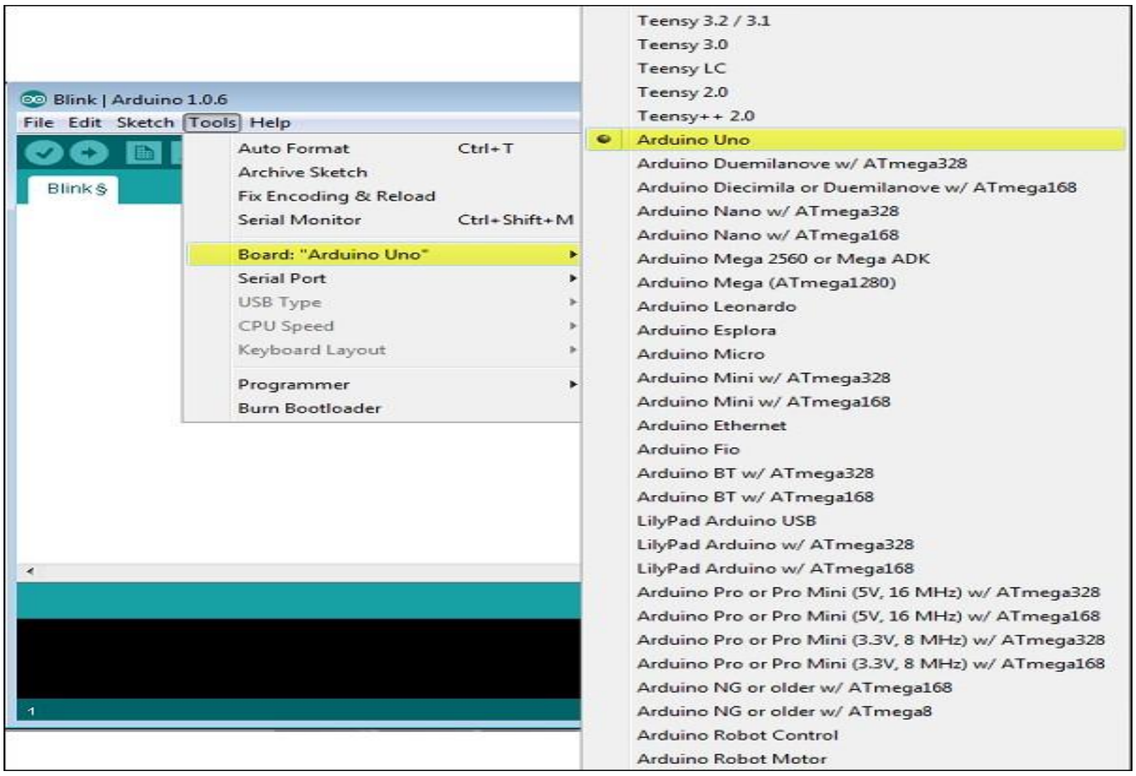


Name	Date modified	Type	Size
drivers	9/27/2015 1:24 PM	File folder	
examples	9/27/2015 1:31 PM	File folder	
hardware	9/27/2015 1:31 PM	File folder	
java	9/27/2015 1:25 PM	File folder	
lib	9/27/2015 1:32 PM	File folder	
libraries	11/19/2015 5:59 PM	File folder	
reference	9/27/2015 1:25 PM	File folder	
tools	9/27/2015 1:25 PM	File folder	
arduino	9/16/2014 3:46 PM	Application	844 KB
arduino_debug	9/16/2014 3:46 PM	Application	383 KB
cygiconv-2.dll	9/16/2014 3:46 PM	Application extens...	947 KB
cygwin1.dll	9/16/2014 3:46 PM	Application extens...	1,829 KB
libusb0.dll	9/16/2014 3:46 PM	Application extens...	43 KB
revisions	9/16/2014 3:46 PM	Text Document	39 KB
rxtxSerial.dll	9/16/2014 3:46 PM	Application extens...	76 KB
uninstall	9/27/2015 1:26 PM	Application	402 KB

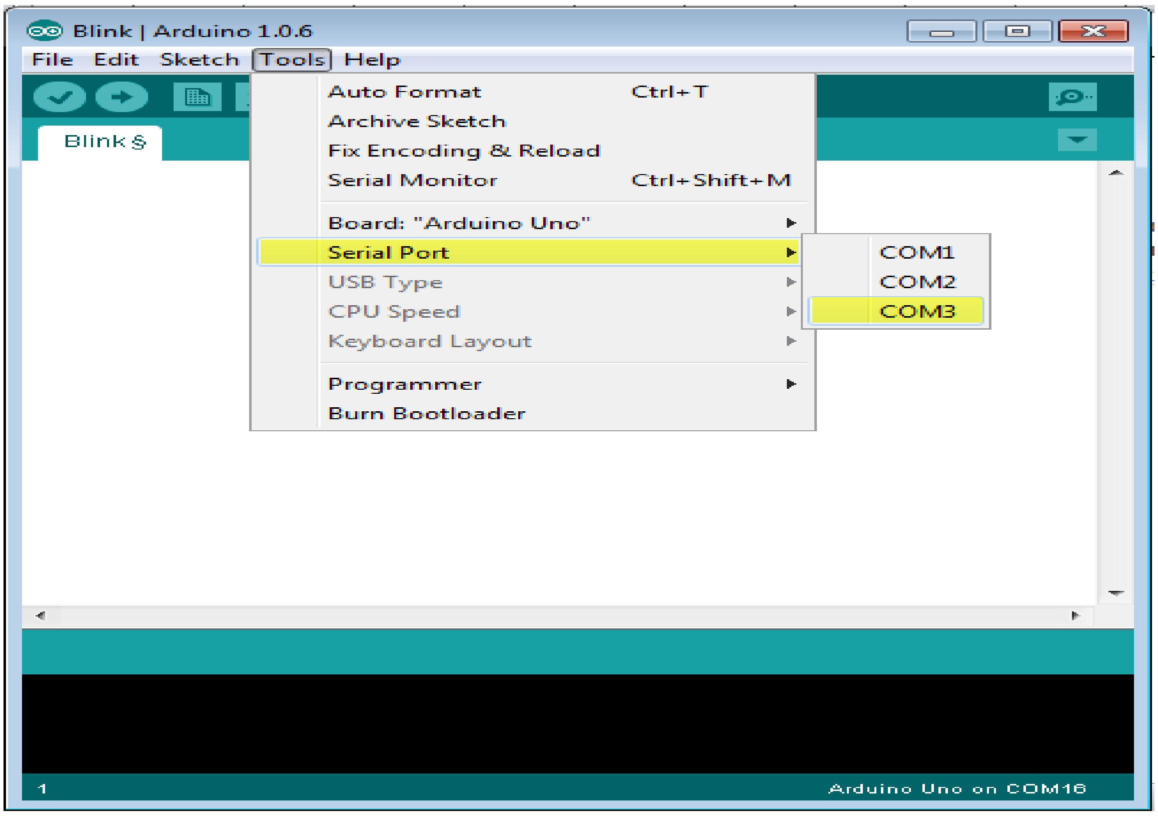
5. **Open your first project.**



6. **Select your Arduino board.**


Blink | Arduino 1.0.6
File Edit Sketch Tools Help
Blink §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board: "Arduino Uno"
Serial Port
USB Type
CPU Speed
Keyboard Layout
Programmer
Burn Bootloader
Teensy 3.2 / 3.1
Teensy 3.0
Teensy LC
Teensy 2.0
Teensy++ 2.0
Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Leonardo
Arduino Esplora
Arduino Micro
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino USB
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
Arduino Robot Control
Arduino Robot Motor
1

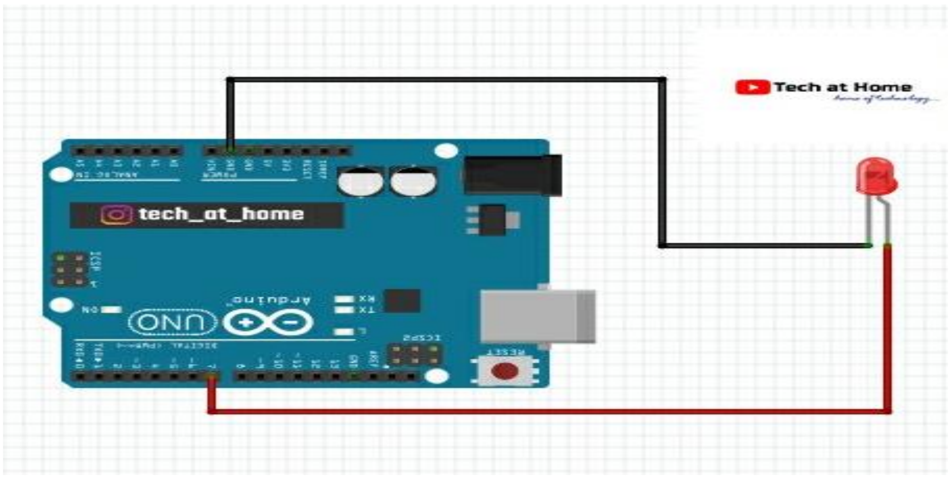
7. Select your serial port.



8. **Upload the program to your board.**

CIRCUIT DIAGRAM:

CODING:



```
int led1=7; void setup()
{

 //pinMode(pin,mode);
 pinMode(7,OUTPUT);
  }

void loop()

{


  digitalWrite(led1,HIGH);
  delay(200);
  digitalWrite(led1,LOW);
  delay(200);
    }
```

Output Screenshot / Image

RESULT:

Thus Turn on and off LED light with Arduino is executed and performed successfully.

EX NO:11 READ A SWITCH, PRINT THE STATE OUT TO THE ARDUINO SERIAL MONITOR

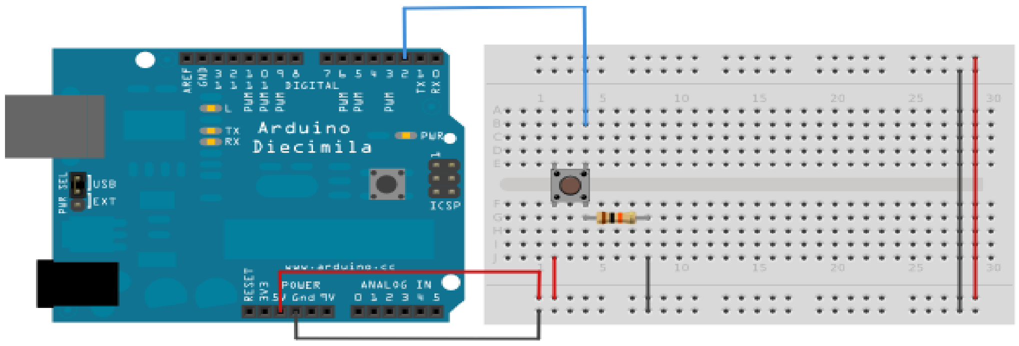
AIM:

To read a switch, print the state out to the Arduino serial monitor.

PROCEDURE:

1. Take a arduino uno board, breadboard, button, jumper wires (male to male) and printer wire (to connect arduino board to PC).
2. Connect the circuit as shown in below figures.
3. Write a program in Arduino uno desktop editor IDE. By opening new sketch.
4. Save file, compile it and upload.

CIRCUIT DIAGRAM:



CODING

```
// digital pin 2 has a pushbutton attached to it. Give it a name:
 int pushButton = 2;
// the setup routine runs once when you press reset: void setup() {
  // initialize serial communication at 9600 bits per second: Serial.begin(9600);
  // make the pushbutton's pin an input:
  pinMode(pushButton, INPUT);
}
```

```
// the loop routine runs over and over again forever:

void loop() {
  // read the input pin:
  int buttonState = digitalRead(pushButton);
  // print out the state of the button:
  Serial.println(buttonState);
  delay(1000);        // delay in between reads for stability
}
```

Output Screenshot / Image

RESULT:

To read a switch, print the state out to the Arduino serial monitor is compiled and executed successfully.

EX NO:12 READ AN ANALOG INPUT AND PRINT THE VOLTAGE TO THE ARDUINO SERIAL MONITOR

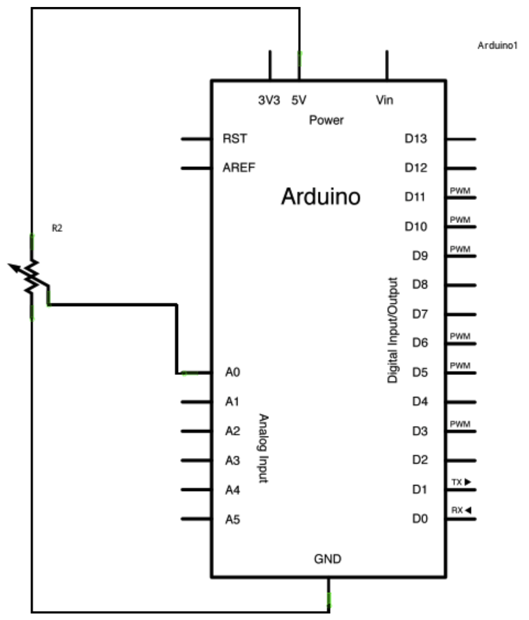
AIM:

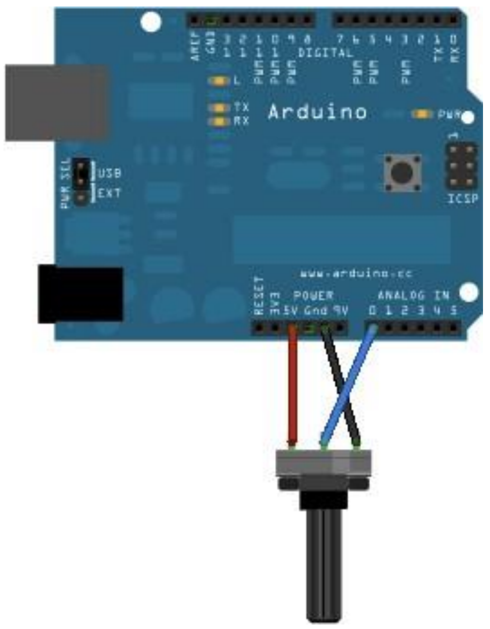
To read an analog input and print the voltage to the Arduino serial monitor.

PROCEDURE:

1. Apparatus is arduino uno board, jumper wires (male to male), printer wire to connect board to PC and 10k potentiometer.
2. Connect centre pin of the potentiometer to A0.
3. Connect outside pin of the potentiometer to +5V
4. Connect GND to GND.
5. Upload the code in Arduino board using Arduino IDE.
6. Open the serial monitor for the output.

CIRCUIT DIAGRAM:





PROGRAM:

```
//Analog Read with Serial Monitor


void setup() {
  //the setup routine runs once when you press reset
  Serial.begin(9600); //initialize serial communication at 9600 bits per second
}
void loop() {
  //the loop routine runs over and over again forever
  int sensorValue = analogRead(A0); //read the input on analog pin 0
  Serial.println(sensorValue); //print out the value you read
  delay(10); //delay in between reads for stability
}
```

Output Screenshot / Image

RESULT:

To read an analog input and print the voltage to the Arduino serial monitor has been executed and performed successfully.

EX NO:13 **DETECT KNOCKS WITH A PIEZO ELEMENT SENSOR**

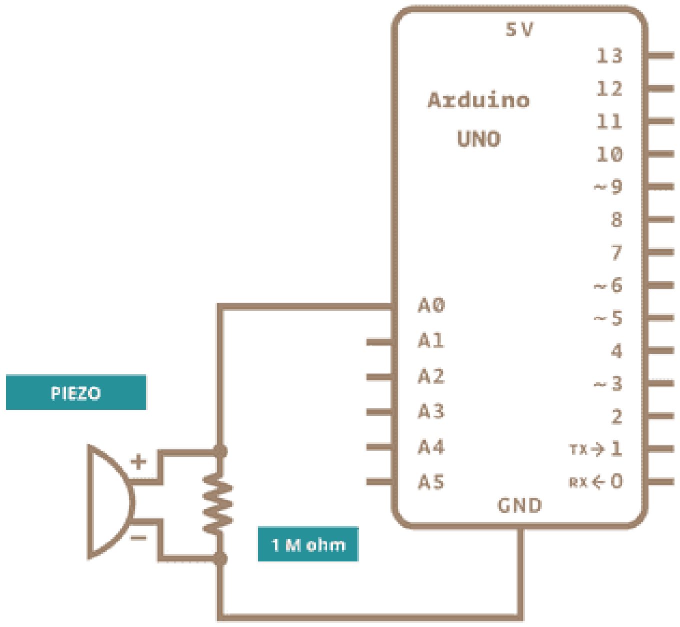
AIM:

To detect knocks with a piezo element sensor using Arduino uno.

PROCEDURE:

1. Apparatus is arduino uno board, jumper wires (male to male), printer wire to connect board to PC and Piezo electric sensor.
2. Connect the circuit,by seeing the Schematic diagram.
3. Connect the USB to the Arduino and start compiling.
4. Run the code and check the output.

CIRCUIT:



CODE:

```
/*

 Knock Sensor
 This sketch reads a piezo element to detect a knocking sound.

 It reads an analog pin and compares the result to a set threshold.

 If the result is greater than the threshold, it writes "knock" to the serial port, and toggles the LED on
 pin 13.
```

The circuit:

- positive connection of the piezo attached to analog in 0
- negative connection of the piezo attached to ground
- 1 megohm resistor attached from analog in 0 to ground

```
// these constants won't change:

const int ledPin = 13;              // LED connected to digital pin 13
const int knockSensor = A0; // the piezo is connected to analog pin 0 const int threshold = 100;
// threshold value to decide when the detected sound is a knock or not

// these variables will change:

int sensorReading = 0;              // variable to store the value read from the sensor pin

int ledState = LOW;                 // variable used to store the last LED status, to toggle the light


void setup() {
  pinMode(ledPin, OUTPUT); // declare the ledPin as OUTPUT Serial.begin(9600);
                                  // use the serial port
}

void loop() {

  // read the sensor and store it in the variable sensorReading: sensorReading =
  analogRead(knockSensor);


  // if the sensor reading is greater than the threshold: if (sensorReading >=
  threshold) {
    // toggle the status of the ledPin:

    ledState = !ledState;

    // update the LED pin itself:

    digitalWrite(ledPin, ledState);

    // send the string "Knock!" back to the computer, followed by newline Serial.println("Knock!");
  }

  delay(100); // delay to avoid overloading the serial port buffer

}
```

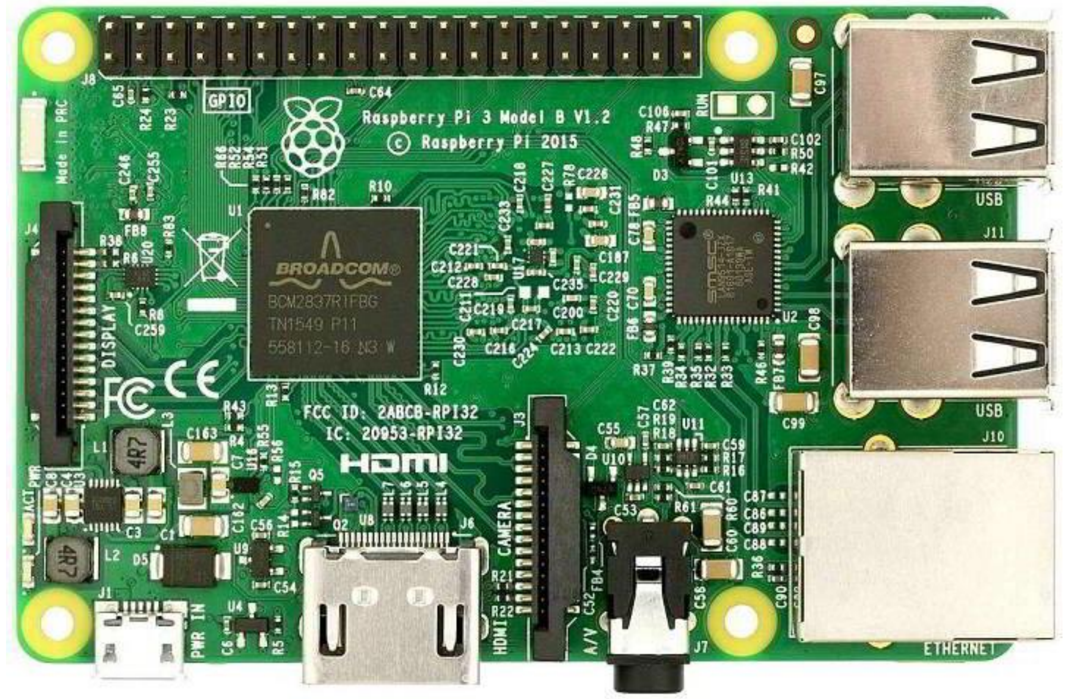
Output Screenshot / Image

RESULT:

To detect knocks with a piezo element sensor using Arduino uno is performed and executed successfully.

EX NO:14 Study and implementation of IoT using Arduino/Raspberry pi

Introduction



Raspberry Pi 3 Board

What is a Raspberry Pi?

Raspberry pi is the name of the "credit card-sized computer board" developed by the Raspberry pi foundation, based in the U.K. It gets plugged into a TV or monitor and provides a fully functional computer capability. It is aimed at imparting knowledge about computing to even younger students at the cheapest possible price. Although it is aimed at teaching computing to kids, can be used by everyone willing to learn to program, the basics of computing, and build different projects by utilizing its versatility.

Raspberry Pi is a small single-board computer. By connecting peripherals like Keyboard, mouse, and display to the Raspberry Pi, it will act as a mini personal computer.

Raspberry Pi is popularly used for real-time Image/Video Processing, IoT-based applications, and Robotics applications.

Raspberry Pi is slower than a laptop or desktop but is still a computer that can provide all the expected features or abilities, at low power consumption.

Raspberry Pi Foundation officially provides Debian based Raspbian OS. Also, they provide NOOBS OS for Raspberry Pi. We can install several Third-Party versions of OS like Ubuntu, Archlinux, RISC OS, Windows 10 IOT Core, etc.Raspbian OS is official Operating System available for free to use. This OS is efficiently optimized to use with Raspberry Pi. Raspbian have GUI which includes tools for Browsing, Python programming, office, games, etc.We should use SD card (minimum 8 GB recommended) to store the OS (operating System).

Raspberry Pi is more than computer as it provides access to the on-chip hardware i.e. GPIOs for developing an application. By accessing GPIO, we can connect devices like LED, motors, sensors, etc and can control them too.It has ARM based Broadcom Processor SoC along with on-chip GPU (Graphics Processing Unit).The CPU speed of Raspberry Pi varies from 700 MHz to 1.2 GHz. Also, it has on-board SDRAM that ranges from 256 MB to 1 GB.Raspberry Pi also provides on-chip SPI, I2C, I2S and UART modules.

There are different versions of raspberry pi available as listed below:

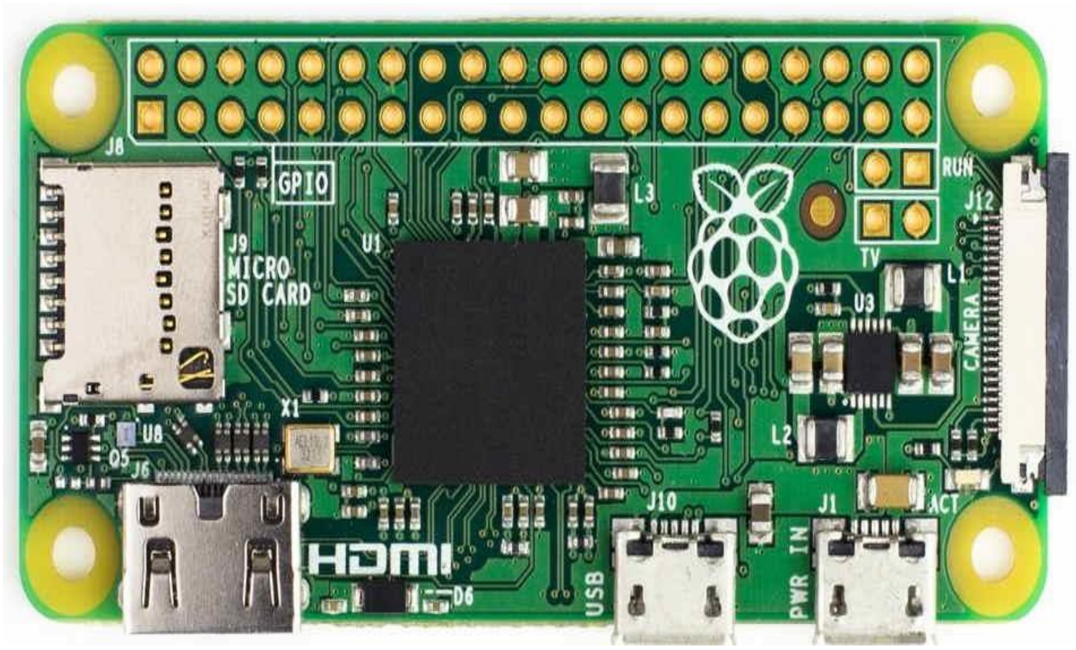
1. **Raspberry Pi 1 Model A**
2. **Raspberry Pi 1 Model A+**
3. **Raspberry Pi 1 Model B**
4. **Raspberry Pi 1 Model B+**
5. **Raspberry Pi 2 Model B**
6. **Raspberry Pi 3 Model B**
7. **Raspberry Pi Zero**

Out of the above versions of Raspberry Pi, more prominently use Raspberry Pi and their features are as follows:

Features	**Raspberry Pi Model B+**	**Raspberry Pi 2 Model B**	**Raspberry Pi 3 Model B**	**Raspberry Pi zero**
SoC	BCM2835	BCM2836	BCM2837	BCM2835
CPU	ARM11	Quad Cortex A7	Quad Cortex A53	ARM11
Operating Freq.	700 MHz	900 MHz	1.2 GHz	1 GHz
RAM	512 MB SDRAM	1 GB SDRAM	1 GB SDRAM	512 MB SDRAM
GPU	250 MHz Videocore IV	250MHz Videocore IV	400 MHz Videocore IV	250MHz Videocore IV
Storage	micro-SD	Micro-SD	micro-SD	micro-SD

Ethernet	Yes	Yes	Yes	No
Wireless	WiFi and Bluetooth	No	No	No

Raspberry Pi zero and Raspberry Pi are shown below



Some Hardware Components shown above are mention below:

1. **HDMI (High-Definition Multimedia Interface):** It is used for transmitting uncompressed video or digital audio data to the Computer Monitor, Digital TV, etc. Generally, this HDMI port helps to connect Raspberry Pi to the Digital television.
2. **CSI Camera Interface:** CSI (Camera Serial Interface) interface provides a connection in between Broadcom Processor and Pi camera. This interface provides electrical connections between two devices.
3. **DSI Display Interface:** DSI (Display Serial Interface) Display Interface is used for connecting LCD to the Raspberry Pi using 15-pin ribbon cable. DSI provides fast High-resolution display interface specifically used for sending video data directly from GPU to the LCD display.
4. **Composite Video and Audio Output:** The composite Video and Audio output port carries video along with audio signal to the Audio/Video systems.

5. **Power LED:** It is a RED colored LED which is used for Power indication. This LED will turn ON when Power is connected to the Raspberry Pi. It is connected to 5V directly and will start blinking whenever the supply voltage drops below 4.63V.
6. **ACT PWR:** ACT PWR is Green LED which shows the SD card activity.

Result:

Thus the various board, function, and components of Raspberry Pi is studied

CONTENT BEYOND SYLLABUS

TRAFFIC LIGHT USING AN ARDUINO UNO.

AIM:

To create a traffic light using an Arduino Uno.

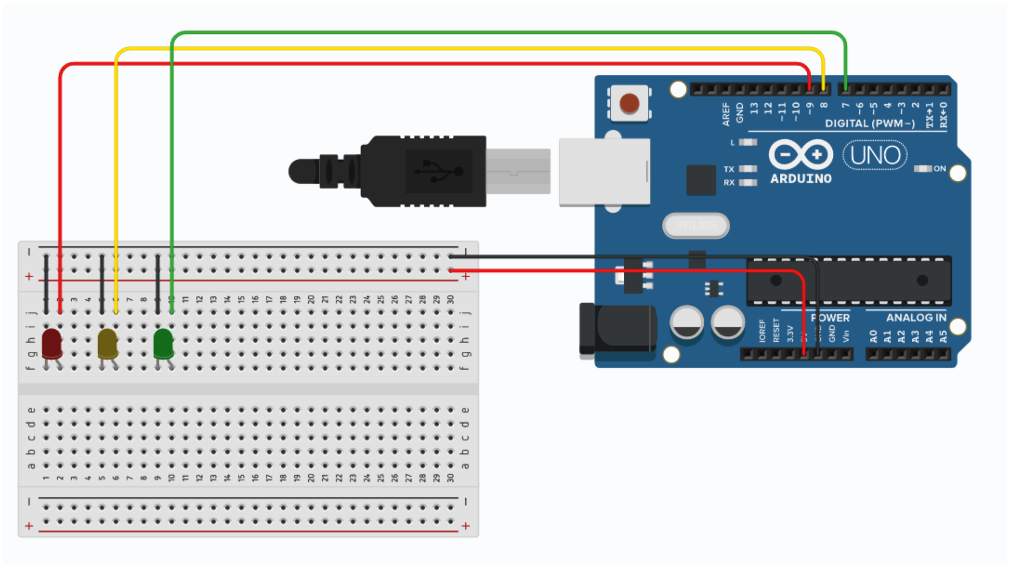
COMPONENTS REQUIRED:

Arduino Uno, SSD1306 OLED, breadboard, jumper wires, LEDs

PROCEDURE:

This simple little project uses an Arduino and some LEDs to replicate a traffic light. It uses code as an internal timer and continues to run until you cut the Arduino's power supply. The LED have been powered by Arduino UNO (Board). It contains a code which uploaded to the board. And once it simulated LED Start's blinking like a traffic light. In this 15 Second will for Red Light 6 Second for Yellow Light (In my Project Blue) and 20 Second for Green Light.

CONNECTIONS:



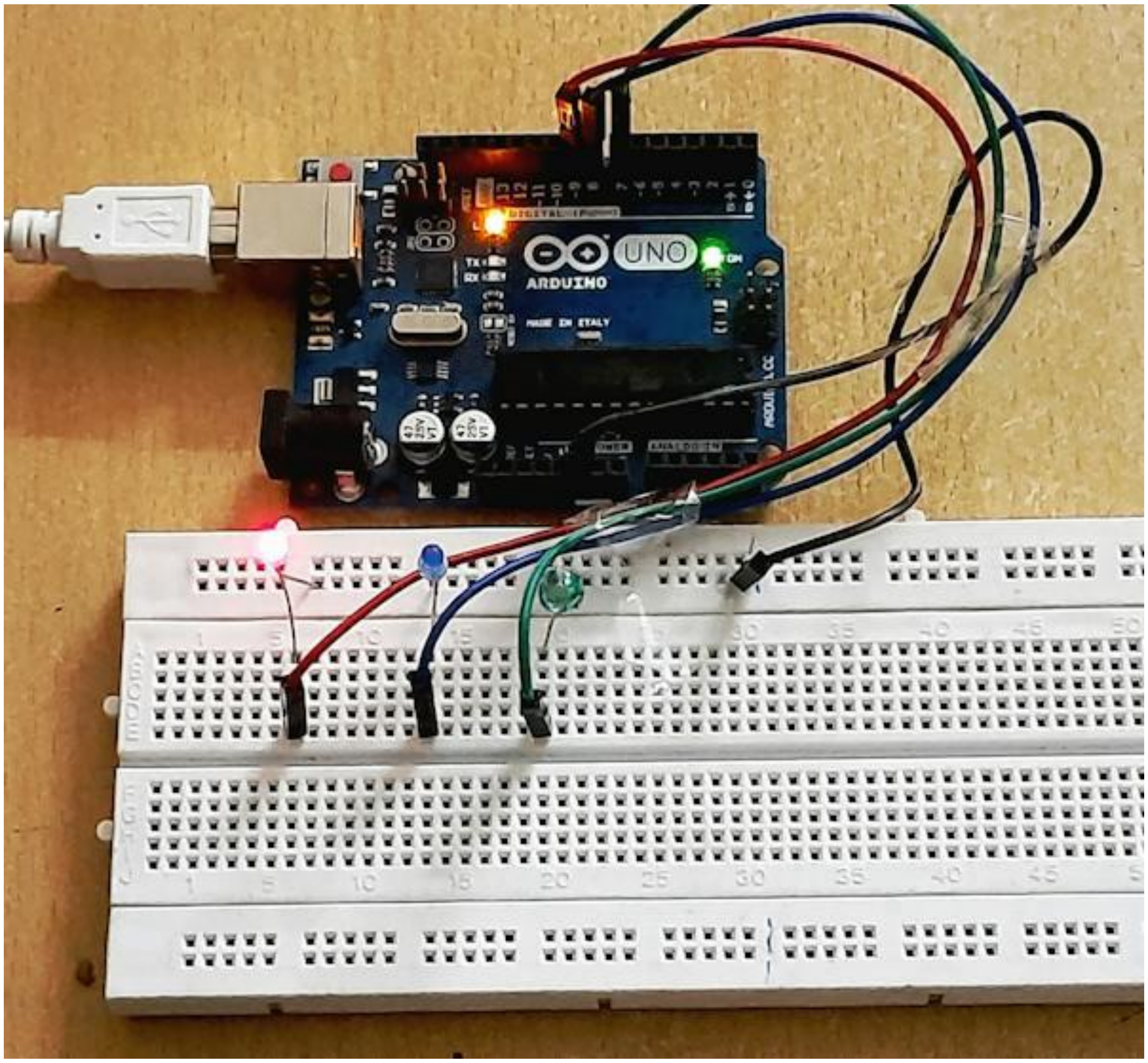
Hookup

- Hook the GND pin (Negative Pin) of all led to Pin GND of Arduino.
- Connect Red LED VCC Pin (Positive Pin) to Pin 9 of Arduino.
- Connect Yellow LED VCC Pin (Positive Pin) to Pin 8 of Arduino.
- Connect Green LED VCC Pin (Positive Pin) to Pin 7 of Arduino.

Code

```
int red = 9;
int yellow = 8;
int green = 7;

void setup(){

  pinMode(red, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(green, OUTPUT);

}
void loop(){
digitalWrite(red, HIGH);
 delay(15000);
digitalWrite(red, LOW);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

digitalWrite(green, HIGH);
delay(20000);
digitalWrite(green, LOW);
//
digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);

  digitalWrite(yellow, HIGH);
delay(1000);
  digitalWrite(yellow, LOW);
delay(500);


}
```

Sample Output Image



RESULT:

Thus the implementation of traffic light signal is successfully done using Arduino

CBS 2: Arduino Based Home Security System Using PIR Sensor

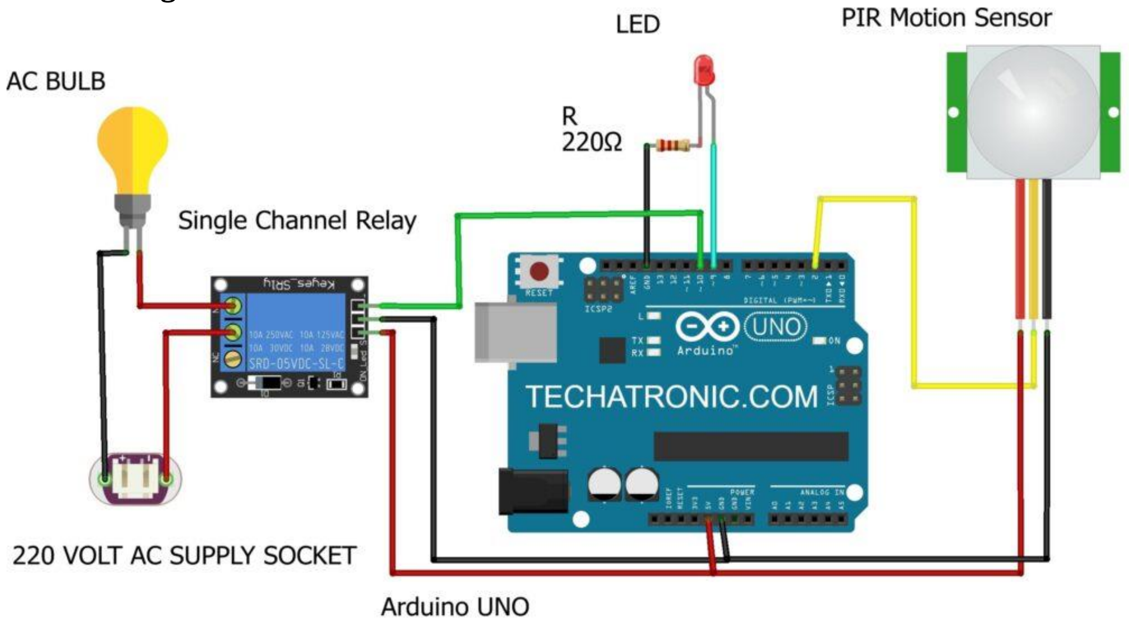
Working

A Passive Infrared Sensor is used to detect the motion of the object that moves in or out of the sensor’s range. It comes with two trim pots for adjusting the sensitivity and the delay time. You can adjust them according to usage. When the sensor detects someone’s movement in its range then HIGH output is generated otherwise LOW output is generated. You can also observe the real-time readings on the Arduino serial monitor as shown below. When the HIGH output is received by Arduino then the LED will turn on and an AC bulb is also turn on. We use a Relay module here for turning the AC bulb on and off.

Components Required

- Arduino UNO
- USB cable for the project
- PIR motion sensor
- Jumper wires
- LED and a 220-ohm resistor
- Relay module
- AC power supply
- AC bulb
- Power supply

Circuit Diagram

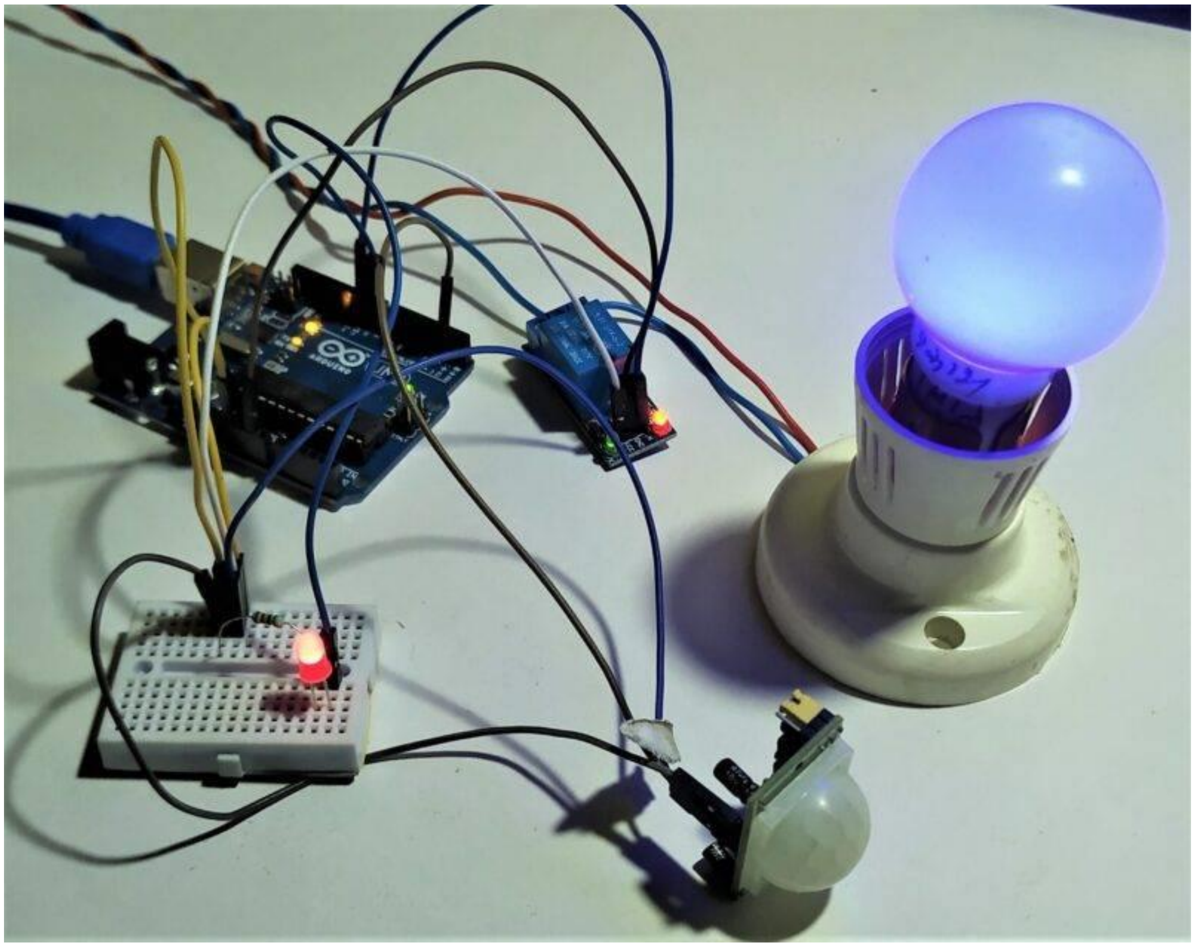


Make the connections according to the circuit diagram given above. Attach the 5-volt pin of the Arduino with the VCC of the PIR motion sensor and the relay module. Connect the GND pin of the Arduino with the GND of the PIR sensor and the relay module. Join the OUT pin of the PIR sensor with the digital-2 pin of the Arduino. Connect the DATA pin of the relay module with the digital-10 pin of the Arduino. On the other side of the relay module make the connections with the AC load and bulb as shown in the diagram. Connect the positive leg of the LED with the digital-9 pin of the Arduino and the negative leg with the GND of the Arduino through a 220-ohm resistor. You can use a breadboard for making the common connections. Now, upload the code and your security system is ready to use.

Program Code

```
int val = 0 ;
 void setup()
 {
 Serial.begin(9600);   // sensor buart rate
 pinMode(2,INPUT);    // pir sensor output pin connected to D2
 pinMode(9,OUTPUT);   // led pin
 pinMode(10,OUTPUT);   // Relay PIN
 digitalWrite(10,HIGH); // Relay Normaly OFF
 }
 void loop()
 {
 val = digitalRead(2); // pir sensor output pin connected
 Serial.println(val); // see the value in serial monitor in Arduino IDE
 delay(100);
 if(val == 1 )
 {
 digitalWrite(9,HIGH);  // LED ON
 digitalWrite(10,LOW);  // Relay ON
 }
 else
 {
 digitalWrite(9,LOW);  // LED OFF
 digitalWrite(10,HIGH); // Relay OFF
 }
 }
```

Output Image



Result

Thus the implementation of alarm system using PIR sensor and Arduino is don successfully.