5.1 Introduction to the Concept of IoT Devices

- IoT devices have unique identity and they are refer as "things" in IoT. Device can perform remote sensing, actuating and monitoring.
- IoT devices can exchange data between them and process data or send to centralized location for processing and storage. Fig. 5.1.1 shows block diagram of IoT device.

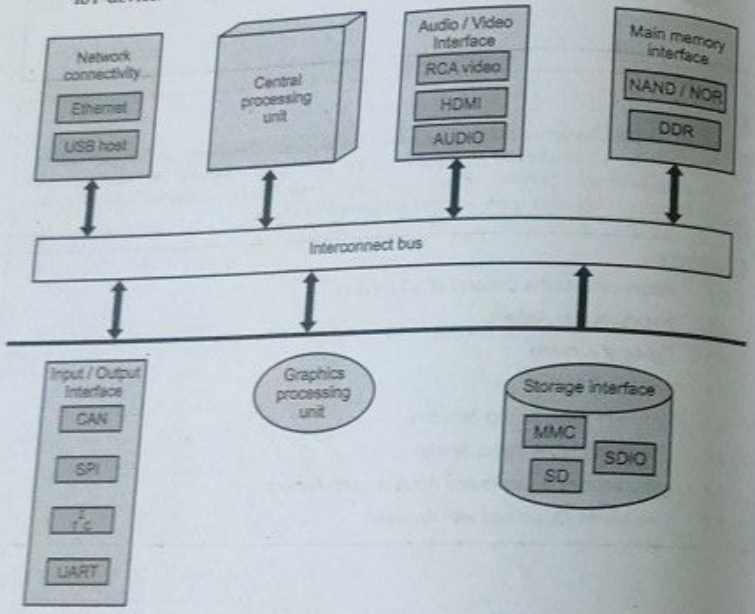


Fig. 5.1.1 : Block diagram of IoT device

- IoT devices provide interface to various wire and wireless devices. Interface includes memory interface, I/O interface for sensors, Internet connectivity interface, storage interface etc.
- Using sensors, IoT collects various information like temperature, light intensity, humidity, air pressure. Some application used cloud based storage. Collected information is stored in cloud and transmitted to other devices.
- Various types of IoT devices are smart clothing, smart watch, wearable sensors, LED lights, automobile industry etc. Fig. 5.1.2 shows IoT devices.

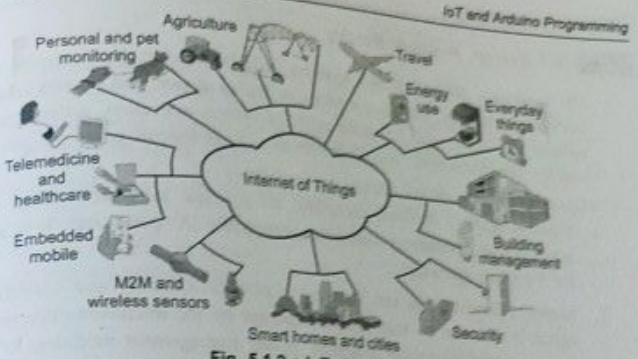


Fig. 5.1.2 : IoT devices

- **Sensor :** Devices that can measure a physical quantity and convert it into a signal, which can be read and interpreted by the microcontroller unit. These devices consist of energy modules, power management modules, RF modules and sensing modules. Most sensors fall into 2 categories : Digital or analog. An analog data is converted to digital value that can be transmitted to the Internet.
- **Actuation :** IoT devices can have various types of actuators attached that allow taking actions upon the physical entities in the vicinity of the device.
- **Communication :** Communication modules are responsible for sending collected data to other device or cloud based servers and receiving data from other devices.
- Analysis and processing modules are responsible for making sense of the collected data.

IoT device life cycle :

- IoT devices are generally more like single-purpose computers. The first life cycle, for example, includes four steps :
  1. **Boot-up :** The device loads the firmware and starts to work as defined.
  2. **Initialization :** Once boot-up is completed, the system reads the configuration, established connections, syncs up data, etc.
  3. **Operation :** The device performs its designed purpose continually.
  4. **Update :** New firmware is installed, the device reboots and then starts to load the new firmware.
- The device should complete its previous life cycle before starting the next life cycle every time the firmware is updated. Eventually, the device will be retired for whatever reason. When it does, it reaches the end of the device life cycle called termination.

5.1.1 IoT System Building Blocks

- The hardware utilized in IoT systems includes devices for a remote dashboard, devices for control, servers, a routing or bridge device, and sensors. These devices manage key tasks and functions such as system activation, action specifications, security, communication, and detection to support-specific goals and actions.
- Major components of IoT devices are as follows :

1. **Control units :** A small computer on a single integrated circuit containing processor core, memory and a programmable I/O peripheral. It is responsible for the main operation.
2. **Sensor :** Devices that can measure a physical quantity and convert it into a signal, which can be read and interpreted by the microcontroller unit. These devices consist of energy modules, power management modules, RF modules, and sensing modules. Most sensors fall into 2 categories : Digital or analog. An analog data is converted to digital value that can be transmitted to the Internet.
   a. Temperature sensors : accelerometers
   b. Image sensors : gyroscopes
   c. Light sensors : acoustic sensors
   d. Micro flow sensors : humidity sensors
   e. Gas RFID sensors : pressure sensors
3. **Communication modules :** These are the part of devices and responsible for communication with rest of IoT platform. They provide connectivity according to wireless or wired communication protocol they are designed. The communication between IoT devices and the Internet is performed in two ways :

   A) There is an Internet-enable intermediate node acting as a gateway;

   B) The IoT Device has direct communication with the Internet.

- The communication between the main control unit and the communication module uses serial protocol in most cases.

4. Power sources : In small devices the current is usually produced by sources like batteries, thermocouples and solar cells. Mobile devices are mostly powered by lightweight batteries that can be recharged for longer life duration.

- **Communication Technology and Protocol :** IoT primarily exploits standard protocols and networking technologies. However, the major enabling technologies and protocols of IoT are RFID, NFC, low-energy Bluetooth, low-energy wireless, low-energy radio protocols, LTE-A, and WiFi-Direct. These technologies support the specific networking functionality needed in an IoT system in contrast to a standard uniform network of common systems.

Working :

1. Collect and transmit data : The device can sense the environment and collect information related to it and transmit it to a different device or to the Internet.
2. Actuate device based on triggers : It can be programmed to actuate other devices based on conditions set by user.
3. Receive information : Device can also receive information from the network.
4. Communication assistance : It provides communication between two devices of same network or different network.

- Fig. 5.1.3 shows working of IoT.

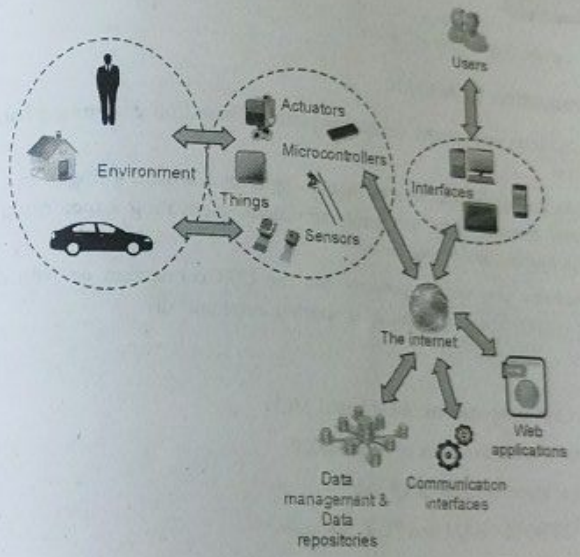


Fig. 5.1.3 : Working of IoT

- Sensors for various applications are used in different IoT devices as per different applications such as temperature, power, humidity, proximity, force etc.
- Gateway takes care of various wireless standard interfaces and hence one gateway can handle multiple techologies and multiple sensors. The typical wireless technologies used widely are 6LoWPAN, Zigbee, Zwave, RFID, NFC etc. Gateway interfaces with cloud using backbone wireless or wired technologies such as WiFi, Mobile, DSL or Fibre.

5.1.2 IoT Devices Versus Computers

IoT devices	Computers
IoT devices are special-purpose devices.	Computers are general-purpose devices.
Limited computational capabilities.	Computational capabilities large.
Limited storage or no storage capabilities.	Unlimited storage capabilities
It is a source of big data.	It manage big data.

Review Question

1. Explain lifecycle of an IoT device.

5.2 Introduction to Arduino

- Arduino is an open-source electronics platform based on easy-to-use hardware and software.
- Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online.
- The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

Features :

- Support fast computations, ARM based MCU
- AVR micro-controller clock is ATSAMSX8I
- Operating input voltages is 3.3 Volt
- It uses EEPROM, SRAM and Flash memory
- It also support USB and UART
- Fig. 5.2.1 shows Arduino board.

Starting clockwise from the top center,

- Analog reference pin (1st pin)
- Digital ground
- Digital pins 2-13 (green)

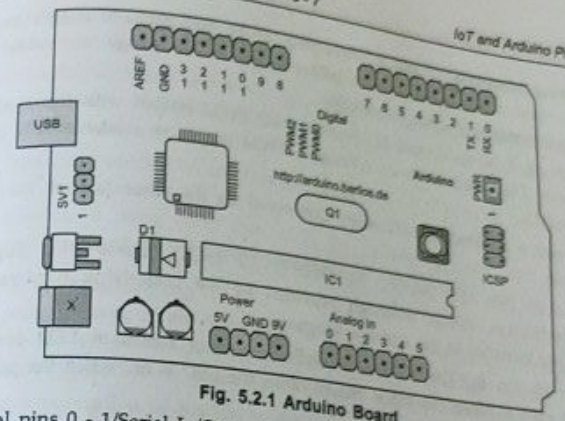


Fig. 5.2.1 Arduino Board

- Digital pins 0 - 1/Serial In/Out - TX/RX : These pins cannot be used for digital I/O (digital Read and digital Write) if you are also using serial communication
- Reset Button - S1
- In-circuit Serial Programmer
- Analog In Pins 0-5
- Power and Ground Pins
- External Power Supply In (9-12VDC) - X1
- Toggles External Power and USB Power (place jumper on two pins closest to desired supply) - SV1 (purple)
- USB

Digital pins

- In addition to the specific functions listed below, the digital pins on an Arduino board can be used for general purpose input and output via the pinMode( ), digitalRead( ), and digitalWrite( ) commands.
- Each pin has an internal pull-up resistor which can be turned on and off using digitalWrite( ), when the pin is configured as an input. The maximum current per pin is 40 mA.
- **Serial :** 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data. On the Arduino Diecimila, these pins are connected to the corresponding pins of the FTDI USB-to-TTL Serial chip. On the Arduino BT, they are connected to the corresponding pins of the WT11 Bluetooth module. On the Arduino Mini and LilyPad Arduino, they are intended for use with an external TTL serial module (e.g. the Mini-USB Adapter).

- **External interrupts :** 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM :** 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the analogWrite() function. On boards with an ATmega8, PWM output is available only on pins 9, 10, and 11.
- **BT reset :** 7. (Arduino BT-only) Connected to the reset line of the bluetooth module.
- **SPI :** 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.
- **LED :** 13. On the Diecimila and LilyPad, there is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Analog pins

- In addition to the specific functions listed below, the analog input pins support 10-bit analog-to-digital conversion (ADC) using the analogRead() function.
- Most of the analog inputs can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19. Analog inputs 6 and 7 (present on the Mini and BT) cannot be used as digital pins.
- **I2C :** 4 (SDA) and 5 (SCL). Support I2C (TWI) communication.

Power pins

- VIN (sometimes labelled "9 V"). The input voltage to the Arduino board when it's using an external power source.
- You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin. Note that different boards accept different input voltages ranges, please see the documentation for your board. Also note that the LilyPad has no VIN pin and accepts only a regulated input.
- **5 V :** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5 V supply.
- **3V3 :** (Diecimila-only) A 3.3 volt supply generated by the on-board FTDI chip.
- **GND :** Ground pins.

Other pins

- **AREF :** Reference voltage for the analog inputs. Not currently supported by the Arduino software.

- **Reset :** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.
- It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.
- The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7 V, however, the 5 V pin may supply less than five volts and the board may be unstable. If using more than 12 V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

Arduino Uno R3 programming

- The programming of an Arduino Uno R3 can be done using IDE software. The microcontroller on the board will come with pre-burned by a boot loader that permits to upload fresh code without using an exterior hardware programmer.
- The communication of this can be done using a protocol like STK500.
- We can also upload the program in the microcontroller by avoiding the boot loader using the header like the In-Circuit Serial Programming.

5.2.1 Features of Arduino Board

- It is an easy USB interface. This allows interface with USB as this is like a serial device.
- The chip on the board plugs straight into your USB port and supports on your computer as a virtual serial port.
- It is easy-to-find the microcontroller brain which is the ATmega328 chip. It has more number of hardware features like timers, external and internal interrupts, PWM pins and multiple sleep modes.
- It is an open source design and there is an advantage of being open source is that it has a large community of people using and troubleshooting it. This makes it easy to help in debugging projects.
- It is a 16 MHz clock which is fast enough for most applications and does not speeds up the microcontroller.
- It has a 32 KB of flash memory for storing your code.

5.3 Types of Arduino

- Arduino board is an open - source platform used to make electronics projects. It consists of both a microcontroller and a part of the software or Integrated Development Environment (IDE) that runs on PC, used to write and upload computer code to the physical board.

- Arduino is an electronic controlling tool to interact with systems. Arduino can read the inputs and gives output according to the program. The controller took the input from the object sensors and light sensor and returned the desired output.
- Types of Arduino are as follows :

1. Arduino UNO

- Arduino UNO is mostly used on Arduino boards among all others. It consists of an Atmega16U2 microcontroller that has a maximum transfer rate and takes enough memory compared to other boards.
- No extra devices are needed for the Arduino UNO board like joystick, mouse, keyboard. The Arduino UNO contain SCL and SDA pins and also have two additional pins fit near to RESET pin.
- It consists of 14-digital I/O pins, where 6-pins can be used as pulse width modulation outputs, 6-analog inputs, a reset button, a power jack, a USB connection, an In-Circuit Serial Programming header (ICSP), etc.
- It includes everything required to hold up the microcontroller; simply attach it to a PC with the help of a USB cable and give the supply to get started with an AC-to-DC adapter or battery.

2. Arduino Leonardo

- This Arduino board uses the Atmega32u4 microcontroller with 20 I/O pins and 16 MHz frequency and acts as a crystal oscillator.
- It is used in a computer system as a mouse or keyboard because it has no additional USB port. It is the cheapest Arduino board on the market.

3. Arduino Mega

- There are 54 pins of which 14 pins are used for PWM output, 16 input pins, and 4 hardware ports, with USB connection, ICSP header, reset pin and a power jack port.
- The microcontroller Atmega2560 consisting of 16 Mhz frequency works as a crystal oscillator. It has a 265 KB flash memory size to store data and it can work fine with the battery as well.

4. Arduino Red Board

- The Arduino Red board uses the mini USB cable for getting programmed and the Arduino IDE is used for this purpose. The RedBoard uses the FTDI FT232RL.
- This board is compatible with Windows operating system and there is no need to change the security settings to make this board working. Red board uses FTDI chip and USB chip for the connection to other device.

5. Arduino Micro

- The Arduino Micro board mainly depends on the ATmega32U4 based Microcontroller that includes 20-sets of pins where the 7-pins are PWM pins, 12-analog input pins.
- This board includes different components like an ICSP header, RST button, small USB connection, crystal oscillator-16 MHz. The USB connection is inbuilt and this board is the shrunk version of the Leonardo board.

6. Arduino Due

- This Arduino board depends on the ARM Cortex-M3 and it is the first Arduino microcontroller board.
- This board includes digital I/O pins - 54 where 12 - pins are PWM o/p pins, analog pins -12, UARTs-4, a CLK with 84 MHz, an USB OTG, DAC-2, a power jack, TWI-2, a JTAG header, an SPI header, two buttons for reset and erase.

5.4 Arduino Toolchain

- A toolchain is a set of programming tools that is used to perform a complex set of operations. In the Arduino Software (IDE) the toolchain is hidden from the user, but it is used to compile and upload the user Sketch. It includes compiler, assembler, linker and Standard C and math libraries.
- A tool chain is simply a set of software tools that work together to complete a task.
- The problem is that a microcontroller can only execute simple, low-level instructions. These simple instructions include things like add variable a to variable b or take variable b and put it in register x. The microcontrollers only speak in binary.
- The USB device refer to connect to your Computer USB and loading with a program called Arduino Sketch IDE. Fig. 5.4.1 shows Arduino Toolchain.

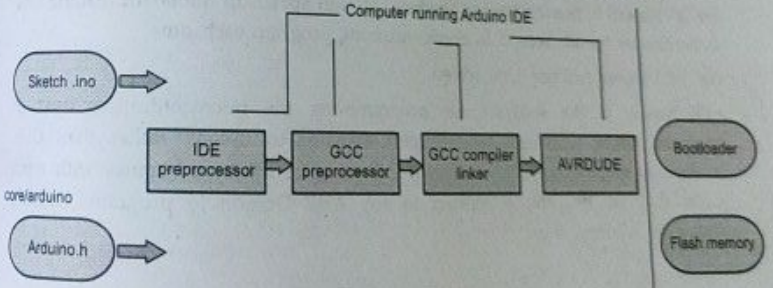


Fig. 5.4.1 Arduino Toolchain

- Sketch code press upload button. The Arduino toolchain is ran to perform the uploading of the code.
- Arduino sketches held the file in .ino extension in a folder with same name.
- Arduino IED starts to perform the following tasks
- Arduino IDE preprocessor assembles the files on the sketch. 1 file is found in the folder.
- As there are many boards with different pins layout, there is a folder /hardware/arduino/variant folder that kept all other type of Arduino pin setup.
- Combining all the files, GCC compiler (which is open source C++ compiler) bundled part of Arduino distribution.
  a) Preprocessor interprets all the #if and #define commands and determines what actually goes into the build.
  b) Next, the code is compiled and linked into a single executable file for type of processor used by board.
  c) After compiler finished, another piece of open source called avrdude actually sends the executable code saved as hexadecimal binary to the board over USB serial interface.
- There is a program called bootloader on Arduino board runs every briefly when Arduino is reset.
- When serial communication starts, the hardware serial link forces a reset to give the bootloader chance to check for any incoming sketches.
- If sketch exist, Arduino programs will unpack the hexadecimal into binary.
- It stores the sketch in the flash memory.

AVR Studio

- Since bootloader removes the needs to have special MCU dependent emulator, but we can directly upload using AVRISPv2, AVRDragon and USBtinyISP to flash to the AVRboard bypassing the bootloader. This can speed up the board loading time as bootloader spends second to check incoming program each time.
- Fig. 5.4.2 shows Arduino Uno screen.
- AVR Studio is the software for programming the microcontrollers used in Arduinos. Using hardware programmer emulator to upload rather than USB. Emulator is more complex than Sketch DE. Most of the board comes with 6-pin header that can be directly connect to any AVR Dragon to programs actually running on ATmega chip.

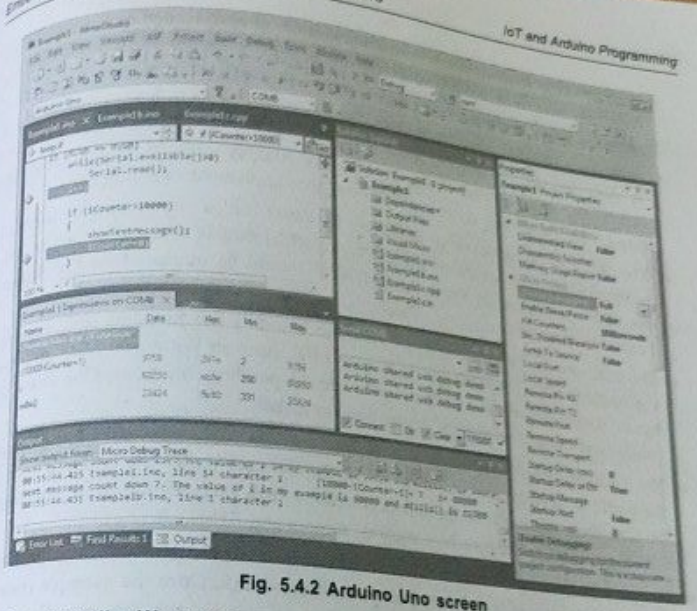


Fig. 5.4.2 Arduino Uno screen

Arduino Compiler Working :

- Enter the compiler. Compiling a program in Arduino is referred to as verifying. The compiler first transforms the code we write into assembly language.
- The assembler, which come with the IDE with the compiler, then translates the assembly language program into machine language. It then creates object files, which combine machine language, data and information it needs to place instructions properly in memory. Often, the assembler creates multiple files which will eventually be put together.
- The linker will take all the independently assembled machine language programs and object files and put them together. This produces a .hex file that the microprocessor can understand and run.

5.5 Arduino Programming Structure

- The Arduino board is connected to a computer via USB, where it connects with the Arduino Development Environment (IDE). The user writes the Arduino code in the IDE, then uploads it to the microcontroller which executes the code, interacting with inputs and outputs such as sensors, motors and lights.

- Arduino code is written in C++ with an addition of special methods and functions. After the sketch is written in the Arduino IDE, it should be uploaded on the Arduino board for execution.
- Libraries : Arduino provides built-in libraries which provide basic functionality. It is also possible to import other libraries and expand the Arduino board capabilities and features. These libraries are roughly divided into libraries that interact with a specific component or those that implement new functions.
- The Arduino programming language is based on a very simple hardware programming language called processing, which is similar to the C language. After the sketch is written in the Arduino IDE, it should be uploaded on the Arduino board for execution.
- The first step in programming the Arduino board is downloading and installing the Arduino IDE. The open source Arduino IDE runs on Windows, Mac OS X and Linux. Downloaded the Arduino software (depending on your OS) from the official website and follow the instructions to install.
- The Arduino board is connected to a computer via USB, where it connects with the Arduino development environment (IDE). The user writes the Arduino code in the IDE, then uploads it to the microcontroller which executes the code, interacting with inputs and outputs such as sensors, motors and lights.
- To get it started with Arduino Uno board and blink the built-in LED, load the example code by selecting **Files>Examples>Basics>Blink**. Once the example code is loaded into IDE, click on the 'upload' button given on the top bar. Once the upload is finished, we should see the Arduino built-in LED blinking. Below is the example code for blinking :

```
// the setup function runs once when we press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the
                                      // voltage level)
    delay(1000);                      // wait for a second
    digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the
                                      // voltage LOW
    delay(1000);                      // wait for a second
}
```


Libraries :

- Libraries are a collection of code that makes it easy to connect to a sensor, display, module, etc. For example, the built-inLiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download.

5.5.1 Sketches

- A program written in the Arduino Programming Language is called sketch. A sketch is normally saved with the ".ino" extension (from Arduino).
- The Arduino program can be divided in three main parts : Structure, Values (variables and constants) and Functions.

1. Structure : setup( ), loop( )
2. Control structure : if, if.....else, for, switch case, while, do....while, break, continue.
3. Variables : Constants → HIGH|LOW, INPUT|OUTPUT, true|false, integer constants, floating point constants, Data types → void, boolean, char, byte, unsigned char
4. Functions : Digital I/O → pinMode( ), digitalWrite( ), digitalRead( ), Analog I/O → analogReference( ), analogRead( ), analogWrite( ).

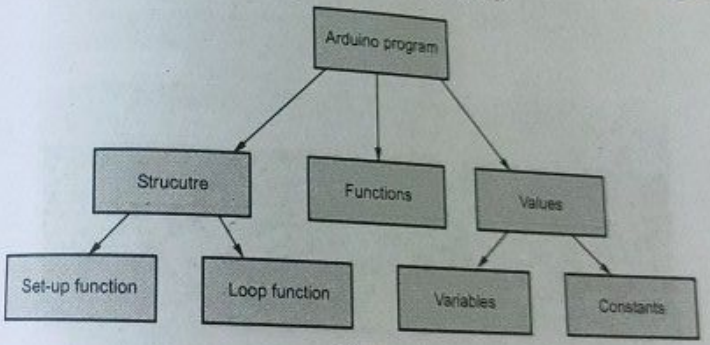


Fig. 5.5.1

- Remarks for writing a program for Arduino :

1. To complete the statement a semicolon ";" is used at the end of the statement.
2. To enclose the block parenthesis "{}" are used. Block in a program contains some statements, declaration of the variables, functions, or loops.

3. Comments can be written for each statement in the code to better understand the statement functionality. It can be done by using double forward slash "//" at the start of the comment if there is only a single line comment. However, if there are multi line comments in a row, a forward slash asterisk "/*" at the start and asterisk forward slash "*/" at the end of the comment. Comments can also be used to exclude any statement.

- Fig. 5.5.2 shows basic Arduino Sketch Structure.

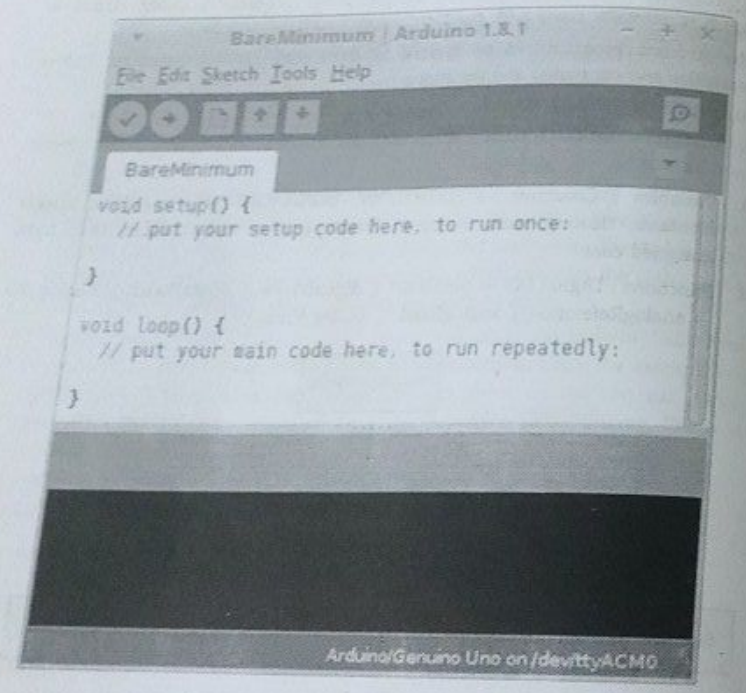


Fig. 5.5.2 Arduino sketch structure

- **Variable declaration section :** In the variable declaration section, usually contains variables that we must declare. To declare is to indicate that something will be used or specified for a particular use.
- **Setup section :** Here we setup the "actions" of what the program is going to do to make the final product function. This could include describing which pins will be

on or off or which mathematical calculations will occur. The setup() function contains initialization of the libraries, variables used for the code. Similarly, pin modes of the Arduino are also declared in this function. It also initializes the communication between the Arduino board and the computer. It only runs once.

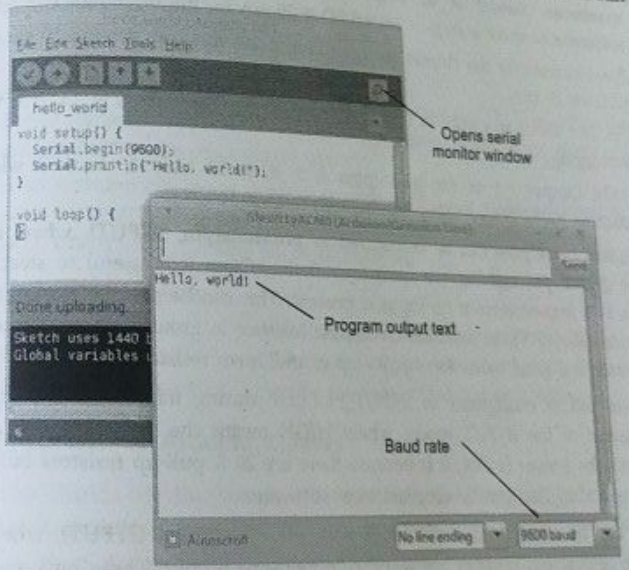
- **Loop section :** The Loop section tells the Arduino to run that set of instructions over and over and makes the final events happen. The loop() function keeps on repeating the instructions and actively controls and monitors the Arduino.

Writing the sketch :

```
void setup() {
  Serial.begin(9600);
  Serial.println("Hello, world!");
}

void loop() {
}
```

- **Running the sketch :** Plug Arduino into PC using a USB cable. Click the Upload button to load the program to the Arduino.
- Now open the Arduino IDE Serial Monitor Window to see the sketch run and print the text message.



- The text that the program outputs should be visible in the serial monitor window.

5.5.2 Pins

- To use the Arduino pins, you need to define which pin is being used and its functionality. A convenient way to define the used pins is by using : '#define pinName pinNumber'
- The functionality is either input or output and is defined by using the pinMode() method in the setup section.
- The pins on the Arduino can be configured as either inputs or outputs. Arduino (Atmega) pins default to inputs, so they don't need to be explicitly declared as inputs with pinMode() when we are using them as inputs.
- Pins configured this way are said to be in a high-impedance state. Input pins make extremely small demands on the circuit that they are sampling, equivalent to a series resistor of 100 M Ω in front of the pin.
- This means that it takes very little current to move the input pin from one state to another and can make the pins useful for such tasks as implementing a capacitive touch sensor, reading an LED as a photodiode, or reading an analog sensor with a scheme such as RCTime.
- Each pin operates at 5 V at HIGH and 0 V at LOW. It can provide or receive a maximum current of 40 mA, but only 20 mA continuous, which is merely sufficient to drive a single - color LED @ 20 mA for full brightness continuously.
- Total current for the chipset shall not exceed 200 mA, i.e., driving 10 single - color LEDs @ 20 mA.
- Pin 13 : There is a built-in LED connected to Pin 13, under Pin 13. It is useful for debugging.
- PWM Output : 6 of the pins (pins 3, 5, 6, 9, 10 and 11, marked with ~) can produce PWM (Pulse Width Modulated) output
- Digital input pins can be configured as **pinMode(pin, INPUT)**, where the pin is the digital pin number we want to initialize. Often it is useful to steer an input pin to a known state if no input is present. This can be done by adding a pull-up resistor (to +5 V), or a pulldown resistor (resistor to ground) on the input. A 10 K resistor is a good value for a pull - up or pulldown resistor.
- If the pin is configured as INPUT_PULLUP during initialization, it inverts the behavior of the INPUT mode, where HIGH means the sensor is OFF and LOW means the sensor is ON. It is because there are 20 K pull-up resistors built into the Atmega chip that can be accessed from software.
- Pins can be configured as OUTPUT with **pinMode(pin, OUTPUT)**, where the pin is the digital pin number we want to initialize as output. These pins are also in a low - impedance state. This means that they can provide a substantial amount of current to other circuits.

digitalWrite()

- Using digitalWrite() function in Arduino IDE, we can write a digital pin, to a HIGH or LOW value. If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value : 5 V for HIGH, 0 V for LOW.
- If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pull-up on the input pin. It is recommended to set the pinMode() to INPUT_PULLUP to enable the internal pull-up resistor.
- The analog input pins can also be used as digital pins, referred to as A0, A1, etc.
- Arduino program for LED blink :

```
int LED =13; // The digital pin to which the LED is connected
void setup ( )
{
  pinMode (LED, OUTPUT); //Declaring pin 13 as output pin
}
void loop( ) // The loop function runs again and again
{
  digitalWrite (LED, HIGH); //Turn ON the LED
  delay(1000); //Wait for 1 sec
  digitalRead (LED, LOW); // Turn off the LED
  delay(1000); // Wait for 1 sec
}
```

- Here, LED is declared globally and is set to pin number 13. This will reduce the number of iterations required to update the pin number in the program when we connect the LED to the other digital pin.
- A pin on Arduino can be set as input or output by using pinMode function.

```
pinMode(13, OUTPUT);      // sets pin 13 as output pin
pinMode(13, INPUT);       // sets pin 13 as input pin
```

Reading/Writing digital values

```
digitalWrite(13, LOW);            // Makes the output voltage on pin 13, 0 V
digitalWrite(13, HIGH);           // Makes the output voltage on pin 13, 5 V
intbuttonState=digitalRead(2);    // reads the value of pin 2 in buttonState
```

5.6 Introduction to Arduino Shields

- Arduino shields are the boards, which are plugged over the Arduino board to expand its functionalities. There are different varieties of shields used for various tasks, such as Arduino motor shields, Arduino communication shields, etc.

- Shield is defined as the hardware device that can be mounted over the board to increase the capabilities of the projects. It also makes our work easy. For example, Ethernet shields are used to connect the Arduino board to the Internet.
- Shields are pieces of hardware that sit on top of Arduino, often to give it a specific purpose. For example, we can use a shield to make it easier to connect and control motors or even to turn Arduino into something as complex as a mobile phone.
- Arduino shields have exactly the same pins as that on the Arduino boards and only in that case the shields can be placed on the Arduino boards. For performing different functions like running some motors or giving access to the internet to the Arduino and many more, there are different shields available in the market.
- There are two main types of shields one is the simple circuit board also called protoshields in which the user solder the components on the shield. The other types of the shields are the ones that come with already built-in components that are used for specific purposes.
- The use of shields is common because they are easily detachable from the Arduino boards; no connecting wires are required to connect the shields to the Arduino. Furthermore, multiple shields can be used with Arduino by mounting them on one on the other and they also make the circuit look neat and compact.
- Arduino shields take all the complexity of the hardware and reduce it to a simple interface. This allows to get an idea up and running fast. Arduino shields also have programming libraries associated with them. These libraries allow us to easily implement the hardware features available on the shield.
- There are shields for all types of things - LCD shields, LED matrix shields, wifi and bluetooth shields, motor shields, power supply shields, counter shields, there are even shields for cooking hot dogs.
- **Ethernet shield :** The Ethernet shields are used to connect the Arduino board to the Internet. We need to mount the shield on the top of the specified Arduino board. The USB port will play the usual role to upload sketches on the board.
- **Proto shield :** Proto shields are designed for custom circuits. We can solder electronic circuits directly on the shield. The shield consists of two LED pads, two power lines and SPI signal pads. The Input Output voltage REFerence (IOREF) and GND are the two power lines on the board.
- The **joystick shield** has all the functions of a modern game controller on a single Arduino-compatible board. It provides not only four pushbuttons to assign to various functions but also a hidden button in the control stick itself. With the ergonomic control stick, we can smoothly transition between x and y axes to perform movements with great accuracy.

- Advantages of using Arduino shields :
  a) It adds new functionalities to the Arduino projects.
  b) The shields can be attached and detached easily from the Arduino board. It does not require any complex wiring.
  c) It is easy to connect the shields by mounting them over the Arduino board.
  d) The hardware components on the shields can be easily implemented.

5.7 Integration of Sensors and Actuators with Arduino

5.7.1 Sensor

- Sensor converts a physical quantity into a corresponding voltage. Sensor is a device that when exposed to a physical phenomenon (temperature, displacement, force, etc.) produces a proportional output signal (electrical, mechanical, magnetic, etc.).
- Fig. 5.7.1 shows sensor.
- Sensors are sophisticated devices that are frequently used to detect and respond to electrical or optical signals. A sensor converts the physical parameter (for example : temperature, blood pressure, humidity, speed, etc.) into a signal which can be measured electrically.

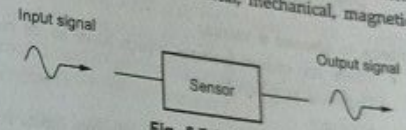


Fig. 5.7.1 Sensor

- The term transducer is often used synonymously with sensors. Sensor is a device that responds to a change in the physical phenomenon.
- On the other hand, a transducer is a device that converts one form of energy into another form of energy. Sensors are transducers when they sense one form of energy input and output in a different form of energy.
- Sensors can also be classified as passive or active. In passive sensors, the power required to produce the output is provided by the sensed physical phenomenon itself whereas the active sensors require external power source.

Specifications of Sensor :

1. **Accuracy :** Error between the result of a measurement and the true value being measured.
2. **Resolution :** The smallest increment of measure that a device can make.
3. **Sensitivity :** The ratio between the change in the output signal to a small change in input physical signal. Slope of the input-output fit line.

4. **Repeatability / Precision :** The ability of the sensor to output the same value for the same input over a number of trials. Fig. 5.7.2 shows accuracy vs resolution.

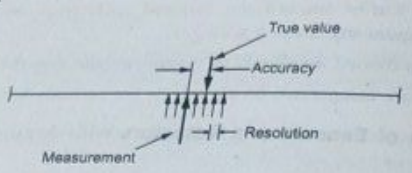


Fig. 5.7.2 Accuracy vs. Resolution

5. **Bandwidth :** The frequency range between the lower and upper cutoff frequencies, within which the sensor transfer function is constant gain or linear.

Criteria to choose a sensor

- There are certain features which have to be considered when we choose a sensor. They are as given below :

1. Accuracy
2. Environmental condition - Usually has limits for temperature/ humidity
3. Range - Measurement limit of sensor
4. Calibration - Essential for most of the measuring devices as the reading's changes with time.
5. Resolution - Smallest increment detected by the sensor
6. Cost
7. Repeatability - The reading that varies is repeatedly measured under the same environment.

- Normally, the output from a sensor requires post processing of the signals before they can be fed to the controller.
- The sensor output may have to be demodulated, amplified, filtered, linearized, range quantized, and isolated so that the signal can be accepted by a typical analog-to-digital converter of the controller.
- Some sensors are available with integrated signal conditioners, such as the microsensors. All the electronics are integrated into one microcircuit and can be directly interfaced with the controllers.

5.7.2 Sensor Component

- Fig. 5.7.3 shows sensor node. A basic sensor node comprises five main components.

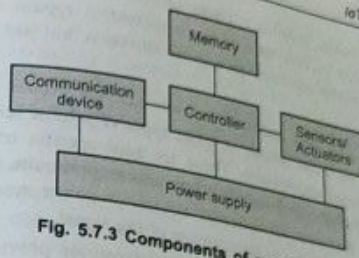


Fig. 5.7.3 Components of sensor node

1. **Controller :** A controller to process all the relevant data, capable of executing arbitrary code.
2. **Memory :** Some memory to store programs and intermediate data; usually, different types of memory are used for programs and data.
3. **Sensors and actuators :** The actual interface to the physical world: devices that can observe or control physical parameters of the environment.
4. **Communication :** Turning nodes into a network requires a device for sending and receiving information over a wireless channel.
5. **Power supply :** As usually no tethered power supply is available, some form of batteries is necessary to provide energy. Sometimes, some form of recharging by obtaining energy from the environment is available as well.

- For actual communication, both a transmitter and a receiver are required in a sensor node. The essential task is to convert a bit stream coming from a microcontroller and convert them to and from radio waves. For practical purposes, it is usually convenient to use a device that combines these two tasks in a single entity. Such combined devices are called **transceivers**.

5.7.3 Sensor Types

- Sensors can be roughly categorized into three categories :

1. Passive, omnidirectional sensors : These sensors can measure a physical quantity at the point of the sensor node without actually manipulating the environment by active probing. In this sense, they are passive. Moreover, some of these sensors actually are self-powered in the sense that they obtain the energy they need from the environment.

- Typical examples for such sensors include thermometer, light sensors, vibration, microphones, humidity, mechanical stress or tension in materials, chemical sensors sensitive for given substances, smoke detectors, air pressure, and so on.

2. *Passive, narrow-beam sensors* : These sensors are passive as well, but have a well-defined notion of direction of measurement. A typical example is a camera, which can "take measurements" in a given direction, but has to be rotated if need be.

3. *Active sensors* : This last group of sensors actively probes the environment, for example, a sonar or radar sensor or some types of seismic sensors, which generate shock waves by small explosions. These are quite specific, triggering an explosion is certainly not a lightly undertaken action and require quite special attention.

- **Active sensors** : Require an external source of power (excitation voltage) that provides the majority of the output power of the signal.
- **Passive sensors** : The output power is almost entirely provided by the measured signal without an excitation voltage.
- Digital sensors : The signal produced or reflected by the sensor is binary.
- Analog sensors : The signal produced by the sensor is continuous and proportional to the measurand.

1. **Temperature sensors :**

- This device collects information about temperature from a source and converts into a form that is understandable by another device or person.
- The best illustration of a temperature sensor is mercury in glass thermometer. The mercury in the glass expands and contracts depending on the alterations in temperature.
- The outside temperature is the source element for the temperature measurement. The position of the mercury is observed by the viewer to measure the temperature.
- There are two basic types of temperature sensors :

a. Contact sensors : This type of sensor requires direct physical contact with the object or media that is being sensed. They supervise the temperature of solids, liquids and gases over a wide range of temperatures.

b. Non-contact sensors : This type of sensor does not require any physical contact with the object or media that is being sensed. They supervise non-reflective solids and liquids but are not useful for gases due to natural transparency. These sensors use Plank's law to measure temperature. This law deals with the heat radiated from the source of heat to measure the temperature.

2. **IR sensor :** This device emits and/or detects infrared radiation to sense a particular phase in the environment. Generally, thermal radiation is emitted by all the objects in the infrared spectrum. The infrared sensor detects this type of radiation which is not visible to human eye.

3. **UV sensor** : These sensors measure the intensity or power of the incident ultraviolet radiation. This form of electromagnetic radiation has wavelengths longer than x-rays but is still shorter than visible radiation. An active material known as polycrystalline diamond is being used for reliable ultraviolet sensing. UV sensors can discover the exposure of environment to ultraviolet radiation.

4. **Touch sensor :** A touch sensor acts as a variable resistor as per the location where it is touched.

5. **Proximity sensor :** A proximity sensor detects the presence of objects that are nearly placed without any point of contact. Since there is no contact between the sensors and sensed object and lack of mechanical parts, these sensors have long functional life and high reliability. The different types of proximity sensors are Inductive proximity sensors, Capacitive proximity sensors, Ultrasonic proximity sensors, photoelectric sensors, Hall-effect sensors, etc.

5.7.4 Actuators

- A device or mechanism capable of performing a physical action. Actuators interact with the world.
- It is output device and convert an electrical signal to a physical output.
- The interface between the microcontroller and the sensors or the actuators is either analog or digital.
- An actuator requires a control signal and a source of energy. An actuator is the mechanism by which a control system acts upon an environment. The control system can be simple, software-based, a human, or any other input.
- When the actuation is a motion, motor have to be used for rotational or linear motion.
- Most of the actuators are guided by software because they turn the control sign into spontaneous action but in the past the actuators are based upon hydraulic, pneumatic, electric, thermal or mechanical means.
- The selection of the proper actuator is more complicated than selection of the sensors, primarily due to their effect on the dynamic behavior of the overall system. Furthermore, the selection of the actuator dominates the power needs and the coupling mechanisms of the entire system.
- Actuators can also be classified as binary and continuous based on the number of stable-state outputs.
- A relay with two stable states is a good example of a binary actuator. Similarly, a stepper motor is a good example of continuous actuator.

- Some types of actuators include Electric motors, Comb drive, Hydraulic cylinder and Pneumatic cylinder.
- A sensor may collect information and route to a control center where a decision is made and a corresponding command is sent back to an actuator in response to that sensed input. Fig. 5.7.4 shows sensor to actuator flow.

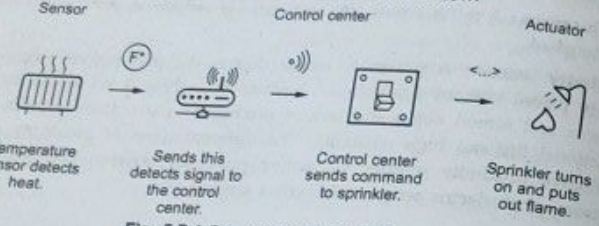


Fig. 5.7.4 Sensor to actuator flow

5.7.5 Difference between Actuator and Sensor

Actuators	Sensor
It is output device.	It is input device.
Convert an electrical signal to a physical output.	Convert a physical parameter to an electrical output.
A component of a machine that is responsible for moving and controlling mechanism.	A device that detects events or changes in the environment and send that information to another electronic device.
It helps to control the environment or physical changes.	It help to monitor the changes in the environment.

5.7.6 Controlling LED by using IR Sensor and Remote

- The IR sensor is a 1838B IR receiver. Whenever a button on the remote is pressed, it will send an infrared signal to the IR sensor in the coded form. The IR sensor will then receive this signal and will give it to the Arduino.
- Whenever a button is pressed on the remote, it sends an infrared signal in encoded form. This signal is then received by the IR receiver and given to the Arduino.
- We will save the code for the buttons that we want to control the LEDs in the Arduino code. Whenever a button on the remote is pressed, the Arduino receives

a code. The Arduino will compare this code with the codes already saved, and if any of them match, the Arduino will turn on the LED connected to that button.

Circuit Diagram

- First, connect the four LEDs to the Arduino. Connect the positives of the four LEDs to the pins 7, 6, 5, and 4. Connect the negative of the four LEDs to GND on the Arduino through the 220 ohm resistors. The longer wires on the LEDs are positive and the shorter wires are negative.
- Then connect the IR sensor to the Arduino. The connections for the IR sensor with the Arduino are as follows :

1. Connect the negative wire on the IR sensor to GND on the Arduino.
2. Connect the middle of the IR sensor which is the VCC to 5 V on the Arduino.
3. Connect the signal pin on the IR sensor to pin 8 on the Arduino.

```
#define first_key 48703
#define second_key 58359
#define third_key 539
#define fourth_key 25979
int receiver_pin = 8;

int first_led_pin = 7;
int second_led_pin = 6;
int third_led_pin = 5;
int fourth_led_pin = 4;
int led[] = {0,0,0,0};
IRrecv receiver(receiver_pin);
decode_results output;

void setup()
{
  Serial.begin(9600);
  receiver.enableIRIn();
  pinMode(first_led_pin, OUTPUT);
  pinMode(second_led_pin, OUTPUT);
  pinMode(third_led_pin, OUTPUT);
  pinMode(fourth_led_pin, OUTPUT);
}

void loop()
{
  if (receiver.decode(&output))
```

```
        {
          unsigned int value = output.value;
          switch(value)
          {
            case first_key:
            if(led[1] == 1) {
            digitalWrite(first_led_pin, LOW);
            led[1] = 0;
          } else {
          digitalWrite(first_led_pin, HIGH);
          led[1] = 1;
          }
      break;
      case second_key:
      if(led[2] == 1) {
                digitalWrite(second_led_pin, LOW);
                led[2] = 0;
              } else {
                  digitalWrite(second_led_pin, HIGH);
                  led[2] = 1;
                  }
    break;
    case third_key:
    if(led[3] == 1) {
              digitalWrite(third_led_pin, LOW);
              led[3] = 0;
              } else {
                  digitalWrite(third_led_pin, HIGH);
                  led[3] = 1;
                }
  break;
  case fourth_key:
  if(led[4] == 1) {
            digitalWrite(fourth_led_pin, LOW);
            led[4] = 0;
          } else {
              digitalWrite(fourth_led_pin, HIGH);
              led[4] = 1;
            }
        break;
        }
```



```
        Serial.println(value);
        receiver.resume();
          }
    }
```

5.7 Reading Switch

- If you have a switch, use the continuity (beeper) function of a digital multimeter (DMM) to understand when the leads are open and when they are connected as the button is pushed.
- The Arduino will read the state of a normally-open push button switch and display the results on the PC using the serial.println() command. You will need a switch, a 10 kohm resistor and some pieces of 22 g hookup wire.
- Create and run this Arduino program

```
void setup()
    {
      Serial.begin(9600);
    }
    void loop()
    {
      Serial.println(digitalRead(3));
      delay(250);
    }
```

5.8 Two Marks Questions with Answers

Q.1 Define sensor.

Ans. : Sensor converts a physical quantity into a corresponding voltage. Sensor is a device that when exposed to a physical phenomenon produces a proportional output signal.

Q.2 What is transducer ?

Ans. : Transducer are devices which convert one type of energy into another, such as electrical energy into light or sound, or creates an electrical output corresponding to some physical parameter.

Q.3 Define Arduino ?

Ans. : Arduino is an open source microcontroller which can be easily programmed, reused and reprogrammed at any instant of

Q.4 What are the types of Arduino board available ?

Ans. :

- *Arduino Ethernet shield : It allows an Arduino board to connect to the internet using the Ethernet library and to read and write an SD card using the SD library.*
- *Arduino Wireless shield : It allows your Arduino board to communicate wirelessly using Zigbee.*
- *Arduino Motor Driver Shield : It allows your Arduino boards to interface with driver of a motor etc.*

Q.5 What are Libraries ?

Ans. : *Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-inLiquidCrystallibrary makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download.*

Q.6 What is basic principle of PWM ?

Ans. : *Pulse width modulation is basically, a square wave with a varying high and low time.*

Q.7 What is variable ?

Ans. : *A variable is used in programming to store a value that may change during the life of the program (or sketch). Memory is set aside for storing the variable and the variable is given a name which allows it to be accessed in the sketch.*

Q.8 What is breadbroad ?

Ans. : *A Breadboard is a helpful tool to build circuits without any soldering. Certain contacts are connected with each other. Therefore it is possible to connect many cables with each other without soldering or screwing them together.*

Q.9 List the example of sensor and transducers.

Ans. : *The examples of sensors are barometer, gyroscope, accelerometer, etc. The thermocouple, thermistor, antennas, etc. are examples of transducers.*

Q.10 What is key difference between sensor and transducer ?

Ans. :

- *The physical changes are sensed by the sensor in the surrounding and intimated in the form of readable quantities to users. On the other hand, a transducer is used for transforming a certain form of energy into a different format.*
- *The sensor does not have any other component for sensing/ processing purpose while a sensor and signal conditioning unit are used for making a transducer work.*

- *The sensor primarily functions to sense physical changes in the environment; on the other hand, a transducer is used for converting physical quantities into electrical signals.*

Q.11 What is Arduino shields ?

Ans. : *Arduino shields are the boards, which are plugged over the Arduino board to expand its functionalities. Shield is defined as the hardware device that can be mounted over the board to increase the capabilities of the projects.*

Q.12 Define sketches

Ans. : *A program written in the Arduino Programming Language is called sketch. A sketch is normally saved with the ".ino" extension (from Arduino).*

Q.13 What is toolchain ?

Ans. : *Toolchain is a set of programming tools that is used to perform a complex set of operations. In the Arduino Software (IDE) the toolchain is hidden from the user, but it is used to compile and upload the user sketch. It includes compiler, assembler, linker and Standard C and math libraries. A tool chain is simply a set of software tools that work together to complete a task*

□□□