SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

19ECT213-IoT SYSTEM ARCHITECTURE

II YEAR/ IV SEMESTER

UNIT 2 – MICROCONTROLLER AND INTERFACING TECHNIQUES FOR IoT DEVICES

TOPIC 3 – EMBEDDED PROGRAMMING

Fundamentals of Arduino Programming

Structure

```
void setup()                    //Preparation function used to declare variables
{                               //First function that runs only one in the
program
        Statement(s);           //used to set pins for serial communication
}
void loop()                     //Execution block where instructions are
executed repeatedly
{                               //this is the core of the Arduino programming
        Statements();           //Functionalities involve reading inputs,
triggering outputs etc.
}
```

Fundamentals of Arduino Programming

setup()

```
void setup()
{
        pinMode(pin, INPUT);        //'pin' configure as input
}
```

loop()

```
void loop()                         //After calling setup(),loop() function does its task
{
        digitalWrite(pin, HIGH);            //sets 'pin' ON
        delay(10000);                       //pauses for ten thousand mili second
        digitalWrite(pin, LOW);             //sets 'pin' OFF
        delay(10000);                       //pauses for ten thousand mili second
}
```

Fundamentals of Arduino Programming

Functions

Syntax: *type functionName(parameters)*

```
{
        Statement(s);
}
```

Example:

```
int delayvar()
{
        int var;                          //create temporary variable var
        var=analogRead(potent);     //read from potentiometer
        var=var/4;                        //convert the value of variable var

        return var;                       //return var
}
```

Fundamentals of Arduino Programming

{ } curly braces

They define beginning and end of function blocks, unbalanced braces may lead to compile errors.

semicolon

It is used to end a statement and separate elements of a program.
Syntax: *int x=14;*

/*…..*/ block comments

Multiline comments begin with /* with a description of the block and ends with */.
Syntax: */*This is an enclosed block of comments*
Use the closing comment to avoid errors/*

Fundamentals of Arduino Programming

//line comments

Single line comment begins with // and ends with next instruction followed.

Syntax: *//This is a Single line comment*

Variables

Example:

int var; //variable 'var' visible to all functions

Data Types

Data type	Syntax	Range
Byte	byte x=100;	0-255
Int	int y=200;	32767 to -32768
Long	long var=8000;	2147483647 to -2147483648
Float	float x=3.14;	3.4028235E+38 to -3.4028235E+38
arrays	int myarray[]={10,20,30,40}	Size depends on the data type associated with declaration.

Operators

Operator	Syntax and its usage

Arithmetic operators
(+,-,/,*)
	x=x+5;
	y-y-6;
	z=z*2;
	p=p/q;

Assignment operators
(=,++,--,+=, =,*=,/=)
	x++; //same as x=x+1
	x+=y; //same as x=x+y
	x-=y; //same as x=x-y
	x*=y //same as x=x*y
	x/=y //same as x=x/y

Comparison operators
(==,!=,<,>,<=,>=)
	x==y //x is equal to y
	x!=y //x is not equal to y
	x<y //x is less than y
	x>y //x is greater than y
	x!=y //x is not equal to y

Logical operators (&&,||,!)	x>2 && x<5 //Evaluates to true only if both expression are true x>2 || y>2 //Evaluates to true if any one expression is true !x>2 //true if only expression is false

Constants

Constants	Usage

TRUE/FALSE	Boolean constants true=2 and false=0 defined in logic levels.
	if(b==TRUE)
	{
	//do something
	}

INPUT/OUTPUT	Used with pinMode () function to define levels. pinMode(13,OUTPUT):
HIGH/LOW	Used to define pin levels HIGH-1, ON, 5 volts LOW-0,OFF, 0 volts digitalWrite(13,HIGH);

Fundamentals of Arduino Programming

Flow control Statements

if

```
if(some_variable == value)
{
  Statement(s);    //Evaluated only if comparison results in a true value
}
```

if…else

```
if(input==HIGH)
{
  Statement(s);    //Evaluated only if comparison results in a true value
}
else
{
  Statement(s);  //Evaluated only if comparison results in a false value
}
```

Fundamentals of Arduino Programming

for

```
for(initialization;condition;expression)
{
    Dosomething;   //Evaluated till condition becomes false
}

for(int p=0;p<5;p++)        //declares p, tests if less than 5, increments by 1
{
digitalWrite(13,HIGH);  //sets pin 13 ON
delay(250);                 // pauses for ¼ second
digitalWrite(13,LOW);  //sets pin 13 OFF
delay(250);                 //pause for ¼ second
}
```

while

```
while(some_variable ?? value)
{
  Statement(s);    //Evaluated till comparison results in a false value
}
```

do...while

```
do
{
Dosomething;
}while(somevalue);
```

Fundamentals of Arduino Programming

Digital and Analog input output pins and their usage

Digital i/o

Methods	Usage

pinMode(pin, mode)	Used in setup() method to configure pin to behave as INPUT/OUTPUT
	pinMode(pin, INPUT) //'pin' set to INPUT
	pinMode(pin, OUTPUT) //'pin' set to OUTPUT

digitalRead(pin)	Read value from a specified pin with result being HIGH/LOW Val=digitalRead(pin); //Val will be equal to input 'pin'
digitalWrite(pin,value)	Outputs to HIGH/LOW on a specified pin. digitalWrite(pin, HIGH); //'pin' is set to HIGH

Example	int x=13; //connect 'x' to pin 13
	int p=7; //connect pushbutton to pin 7
	int val=0; //variable to store the read value
	void setup()
	{
	pinMode(x,OUTPUT); //sets 'x' as OUTPUT
	pinMode(p,INPUT); //sets 'p' as input
	}
	void loop()
	{
	val=digitalRead(p); // sets 'value' to 0
	digitalWrite(x,val); //sets 'x' to button value
	}

Fundamentals of Arduino Programming

Analog i/o

Methods	Usage
analogRead(pin)	Reads value from a specified analog pin works on pins 0-5. val=analogRead(pin); //'val' equal to pin
analogWrite(pin,value)	Writes an analog value using pulse width modulation (PWM) to a pin marked PWM works on pins 3, 5, 6,9,10.

Example	int x=10; //connect 'x' to pin 13
	int p=0; //connect potentiometer to analog pin 7
	int val; //variable for reading
	void setup() { } // No setup is needed
	void loop()
	{
	val=analogRead(p); // sets 'value' to 0
	val/=4;
	analogWrite(x,val); //outputs PWM signal to 'x'
	}

Fundamentals of Arduino Programming

time

Methods	Usage
delay(ms)	Pauses for amount of time specified in milliseconds. delay(1000); //waits for one second
millis()	Returns the number of milliseconds since Arduino is running. val=millis(); //'val' will be equal to millis()

math

Methods	Usage
min(x,y)	Calculates minimum of two numbers val=min (val,10); //sets 'val' to smaller than 10 or equal to 10 but never gets above 10.
max(x,y)	val=max(val, 10); // sets 'val' to larger than 100 or 100.

random

Methods	Usage
randomSeed(value)	Sets a value/seed as starting point for random () function.
random(min,max)	Allows to return numbers within the range specified by min and max values. val=random(100,200); //sets 'val' to random number between 100-200

Example	int rnumber; // variable to store random value
	int x=10;
	void setup()
	{
	randomseed(millis()); //set millis() as seed
	rnumber=random(200); //random number from 0-200
	analogWrite(x,rnumber); //outputs PWM signal
	delay(500);
	}

Serial

Methods	Usage

Serial.begin(rate)	Opens serial port and sets the baud rate for serial data transmission.
	void setup()
	{
	Serial.begin(9600); //sets default rate to 9600 bps
	}

Serial.println(data)	Prints data to the serial port Serial.println(value); //sends the 'value' to serial monitor

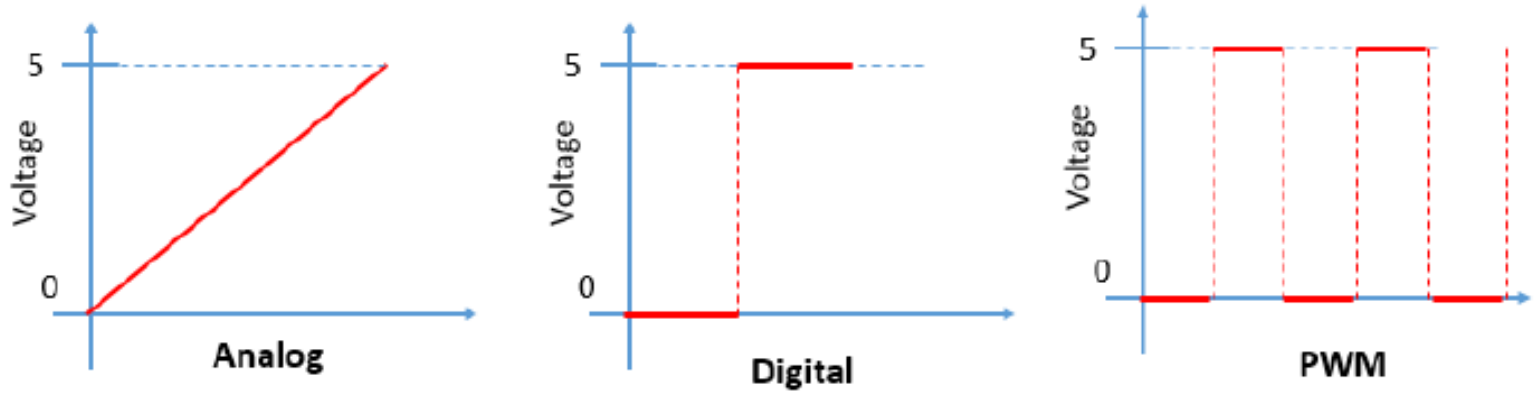
Difference between Analog, Digital and PWM Pins

In **analog pins,** you have unlimited possible states between 0 and 1023. This allows you to read sensor values. For example, with a light sensor, if it is very dark, you'll read 1023, if it is very bright you'll read 0 If there is a brightness between dark and very bright you'll read a value between 0 and 1023.

In **digital pins**, you have just two possible states, which are on or off. These can also be referred as High or Low, 1 or 0 and 5V or 0V. For example, if an LED is on, then, its state is High or 1 or 5V. If it is off, you'll have Low, or 0 or 0V.

PWM pins are digital pins, so they output either 0 or 5V. However these pins can output "fake" intermediate voltage values between 0 and 5V, because they can perform "Pulse Width Modulation" (PWM). PWM allows to "simulate" varying levels of power by oscillating the output voltage of the Arduino.

Fundamentals of Arduino Programming

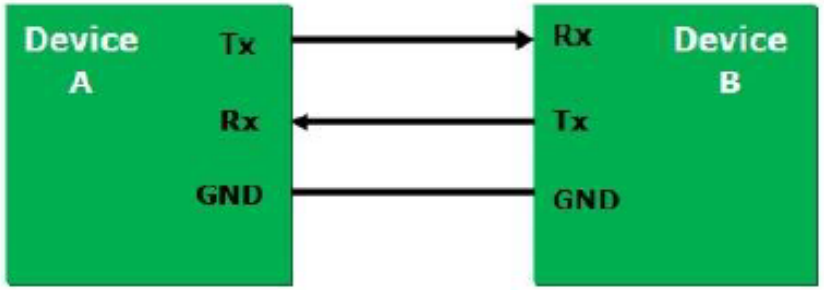


Difference between Analog, Digital and PWM Pins

Introduction to Communications

Serial (UART) communications:

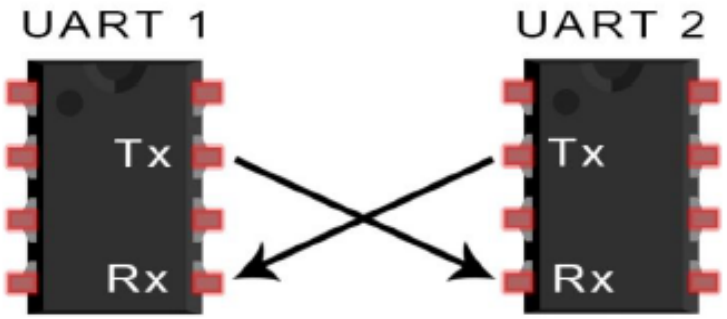
- Serial communication on Arduino pins Tx/Rx uses TTL logic levels which operates at either 5V/3.3V depending the type of the board used.
- Tx/Rx pins should not be connected to any source which operates more than 5V which can damage the Arduino board.
- Serial communication is basically used for communication between Arduino board and a computer or some other compatible devices.



Introduction to Communications

Serial (UART) communications:

- Every Arduino board will have at least one serial port known as UART.
- Serial communicates on digital pins Rx(pin 0) and Tx(pin 1) with the computer via USB, pin 0 and pin 1 cannot be used for digital input or output.
- The built in serial monitor can be used to communicate with an Arduino board by selecting same baud rate that is used in the call to begin () which will come across in the later part of the chapter



Introduction to Communications

SPI communications

- Serial communication Interface (SPI) is a synchronous data protocol used by large microcontrollers for communicating with one or more peripheral devices for a shorter distance and also used for communication between two devices.
- With SPI there will be always one master device which is a microcontroller like Arduino which controls the functionalities of other peripheral devices.
- Devices have three lines in common which are as follows
  - MISO (Master in Slave Out)- Slave line for sending data to the master.
  - MOSI (Master Out Slave In)- Master sending data to peripherals
  - SCK Serial clock) - clock pulses which synchronize data transmission generated by the master And one of the specific line for every device is
  - SS (slave select) - pin on each device that the master can use to enable and disable specific devices.

Introduction to Communications

SPI communications

- When device SS pin is low, communication happens with the master, if SS pin is high device ignores the maser. This allows multiple SPI devices sharing the the same MISO, MOSI and CLK lines.
- To program a new SPI device some key points to be noted which are
  - Maximum SPI speed of the device used?
  - How data is shifted like MSB/LSB?
  - Data clock is idle when high/low.