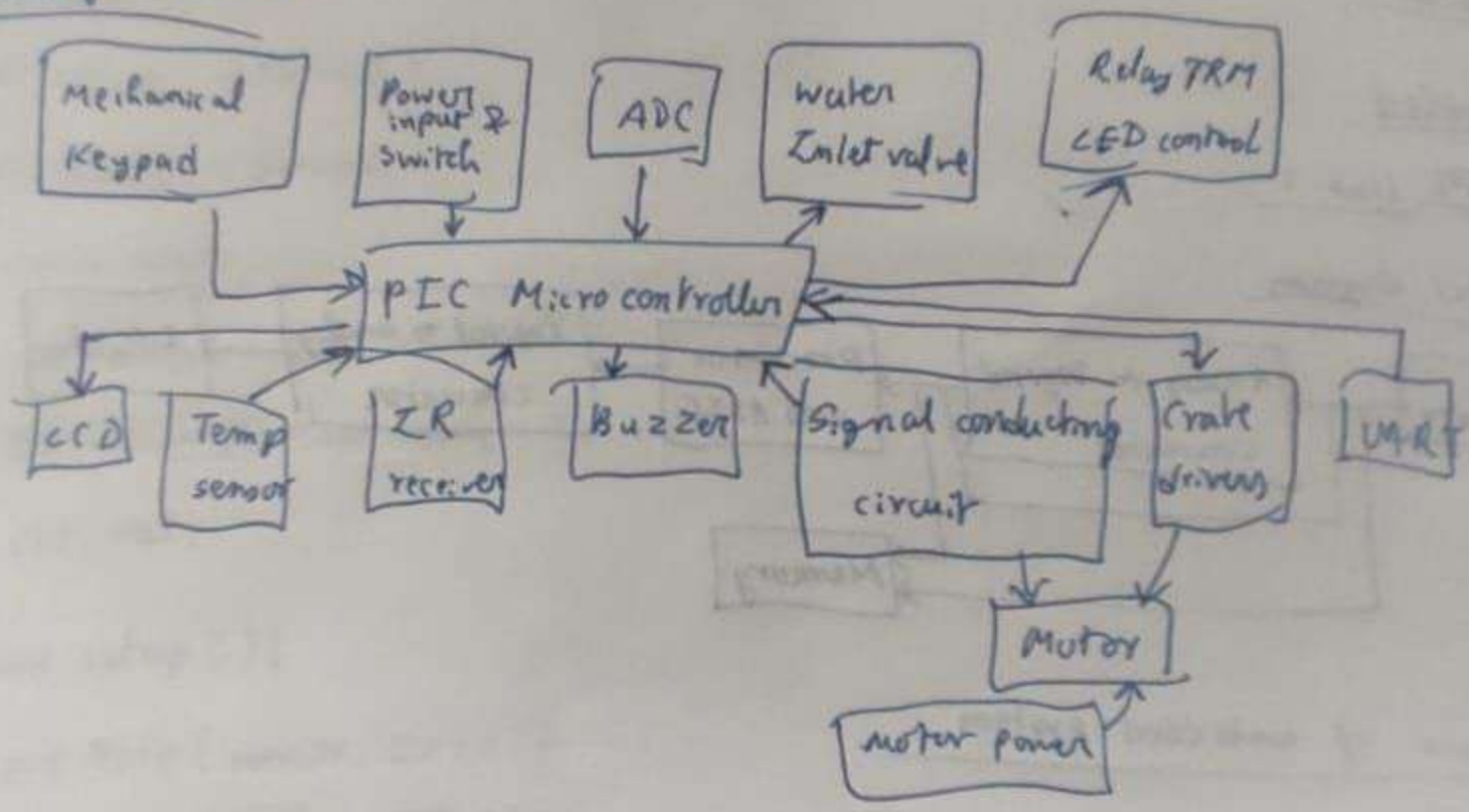
Washing Machine



1) Program selection

2) Sensor monitoring

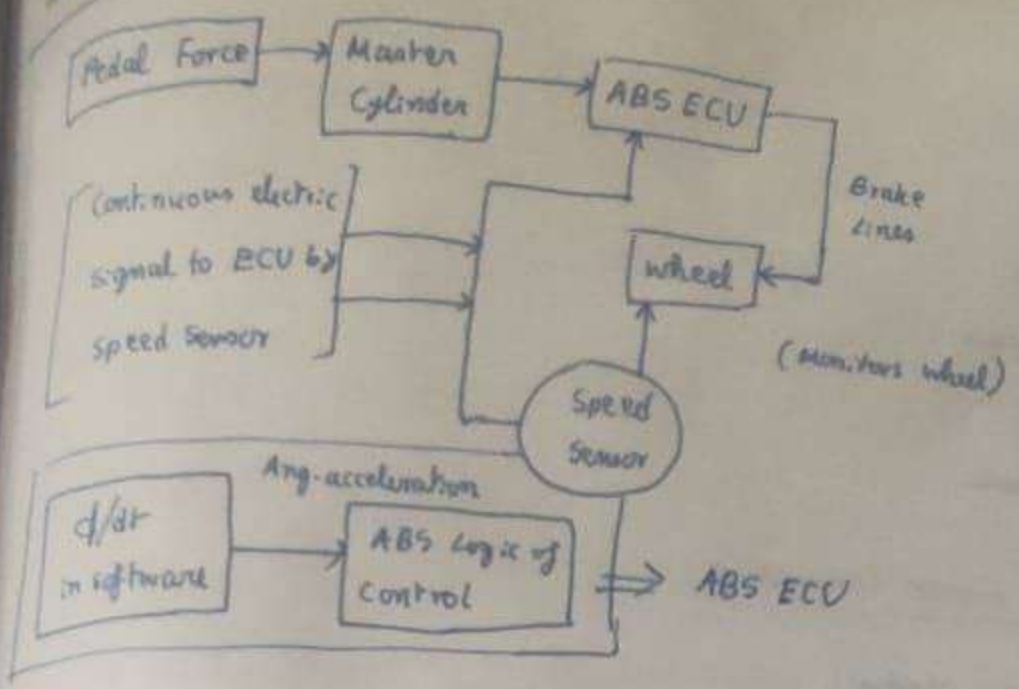
3) Control Algorithm Execution

4) User feedback and display

5) Safety features

6) Power management

Anti lock Brake System



It has :- i) wheel speed sensor

ii) Toothed sensor ring

iii) ECU

iv) Hydraulic modulator

v) Vaccum booster

vi) Hydraulic lines

vii) Wiring from ECU

viii) Master cylinder with proportioner valve.

Speed sensors are used to calculate the acceleration & deceleration of wheel.

When the wheel of vehicle rotates, it induces magnetic field around the sensor. The fluctuation in this magnetic field generates voltage in sensor. This is sent to controller and it reads acceleration & deceleration.

Each brake line is controlled by the ABS has a valve. It works in 3 pos.

Pos 1 → valve open

Pos 2 → valve blocks the line & separates brake from master cylinder

Pos 3 → some extra pressure released by valve

Pump is used to restore the pressure to the hydraulic brakes after valve releases pressure. When the controller detects wheel slip, it sends signals to release the valve.

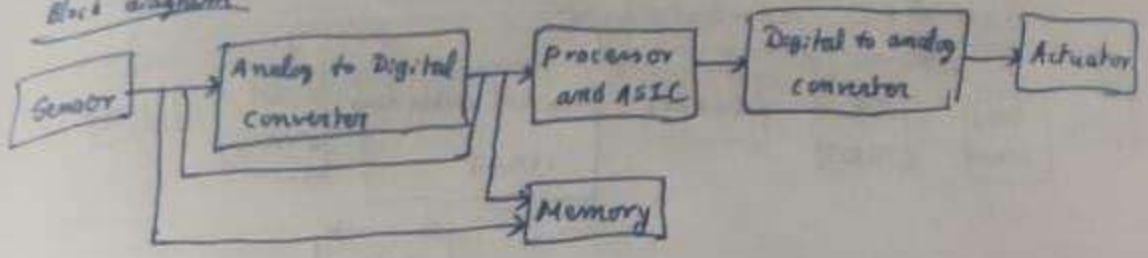
Controller (ECU) is receiving the information from each individual wheel speed sensors and if wheel loses its traction with ground, alarm signal is sent to controller. It limits brake force and activates ABS modulator, which actuates braking valves & varies pressure

1 SEE

Embedded

~~SEE~~ Unit-1

Block diagram

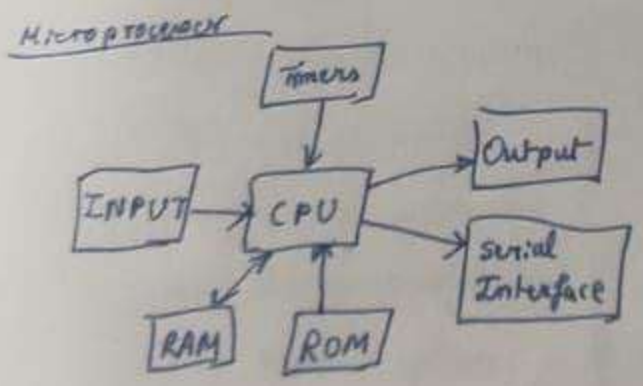


Structure of embedded system

:) Hardware

:) Software & firm ware

:) Real Time Operating System



Keypad code

```
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
    {'1', '2', '3', 'A'},
    {'4', '5', '6', 'B'},
    {'7', '8', '9', 'C'},
    {'*', '0', '#', 'D'}
    };
byte rowPins[ROWS] = {13, 4, 11, 10}; keypad
byte colPins[COLS] = {9, 8, 7, 6};

void setup() {
for (byte i = 0; i < ROWS; i++){
pinMode(rowPins[i], INPUT_PULLUP);}

for (byte i = 0; i < COLS; i++){
 pinMode(colPins[i], OUTPUT); }

 Serial.begin(9600); }

void loop(){
 char key = getKey();
 if(key) { Serial.println(key); }

 char getKey(){
   static char lastKey = 0;
    char currentKey = 0;
   for (byte c = 0; c < COLS; c++){
   digitalWrite(colPins[c], LOW); }
    currentKey = keys[r][c]; } }
   digitalWrite(colPins[c], HIGH); }
if (currentKey != lastKey){
 delay(50); lastKey = currentKey;
 return currentKey; }
```

return 0; }

Unit - 2

Key components of IDE :-

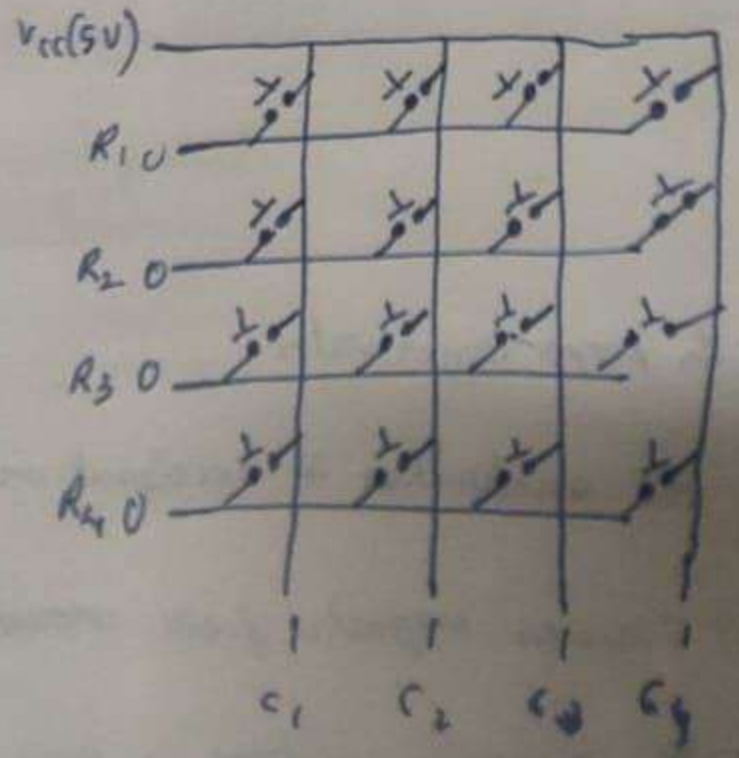
1) Code Editor
2) Library Manager
3) Compiler & Uploader
4) Serial Monitor
5) Integrated examples

Structure of Arduino program (Embedded C) :-

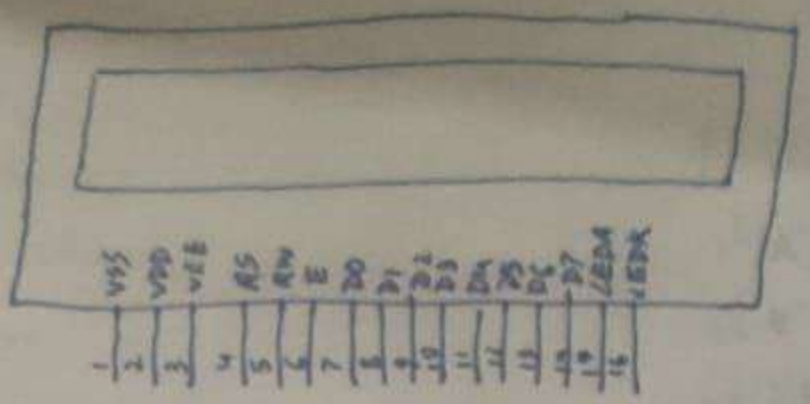
setup() Function :- 1) setup()
2) loop()

4 x 4 Keypad

Structure



16x2 LCD



- VCC → +5V

  ⊖GND → GND

  VEE → middle pin of potentiometer

  RW → GND

  RS → digital pin

  E → digital pin

  D4-D7 → digital pins

Functions → lcd.begin(); lcd.print(); lcd.clear(), lcd.setCursor();

Temperature Conversion

```
void setoup(){
  Serial.begin(9600); }

void loop(){
  float celsius, farenheit;
  Serial.println("Enter Temp in Celsius");
  while(!Serial.available());
  celsius = serial.parseFloat();
  farenheit = (celsius * 9.0 / 5.0)+32;
  serial.print("Temperature in Farenheit: ");
  Serial.println(farenheit); }
```

LED Brightness with Potentiometer

```
int potPin = A0;
int led = 9;

void setup (){
pinMode (led, OUTPUT);
pinMode (potPin, INPUT);
Serial.begin(9600); }

void loop (){
int potValue = analogRead(potPin);
int brightness = map(potValue, 0,1023, 0,255);
analogWrite ( led, brightness);
Serial.print ("Potentiometer Value: ");
Serial.print(potValue);
Serial.print (", Brightness: ");
Serial.println (brightness);
delay (100);
}
```

Arithmetic operators

+, −, *, /, % (remainder) (modulus)

Logical Ops

And → &&
OR → ||
NOT → !

Comparison OPS

Equal to → ==
Not equal to → !=
Greater than → >
Less than → <
Greater than or equal to → >=

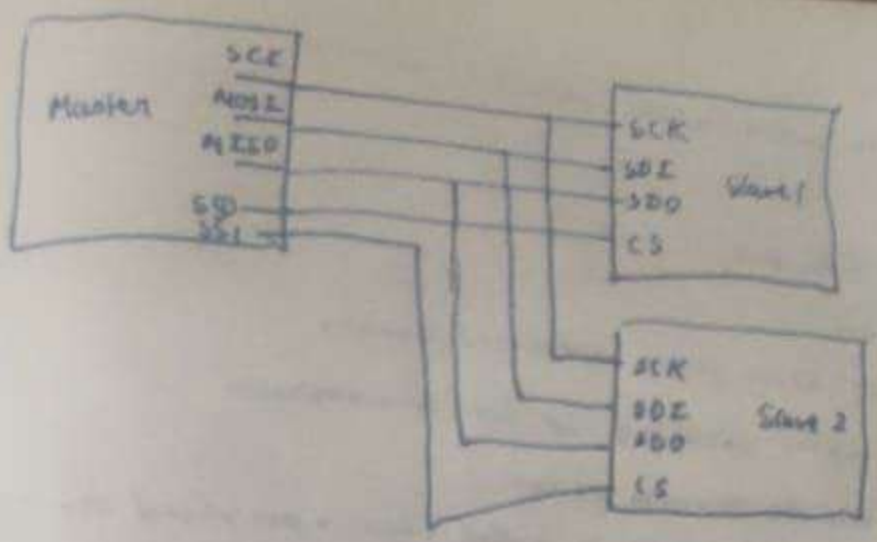
Assignment Ops

Assignment → =
Addition assignment → +=
Subtraction → -=
Multiplication → *=
Division → /=
Modulus → %=

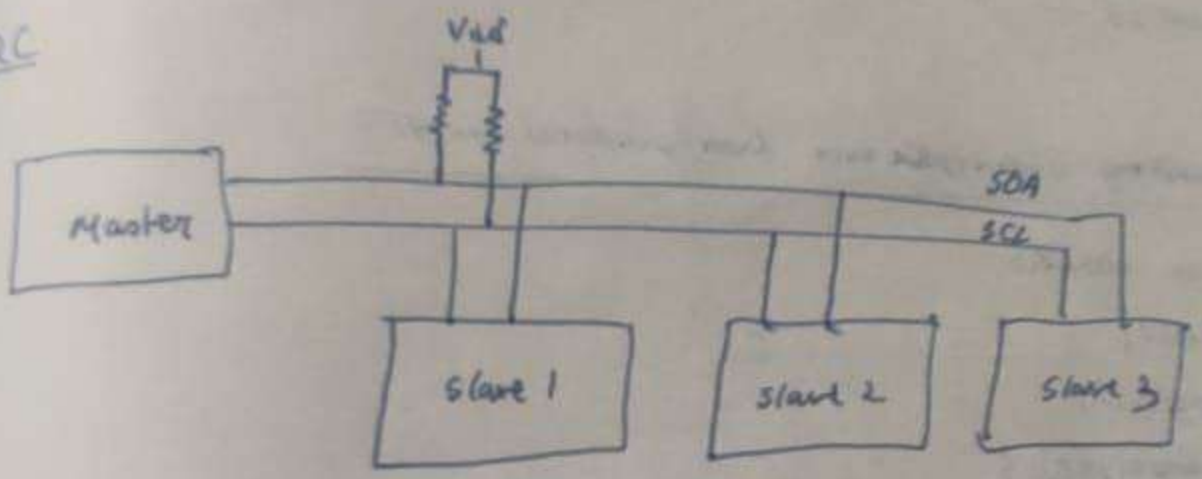
Bitwise Ops

Bitwise And → &
" OR → |
" XOR → ^
" NOT → ~
Left shift → <<
Right shift → >>

SPI

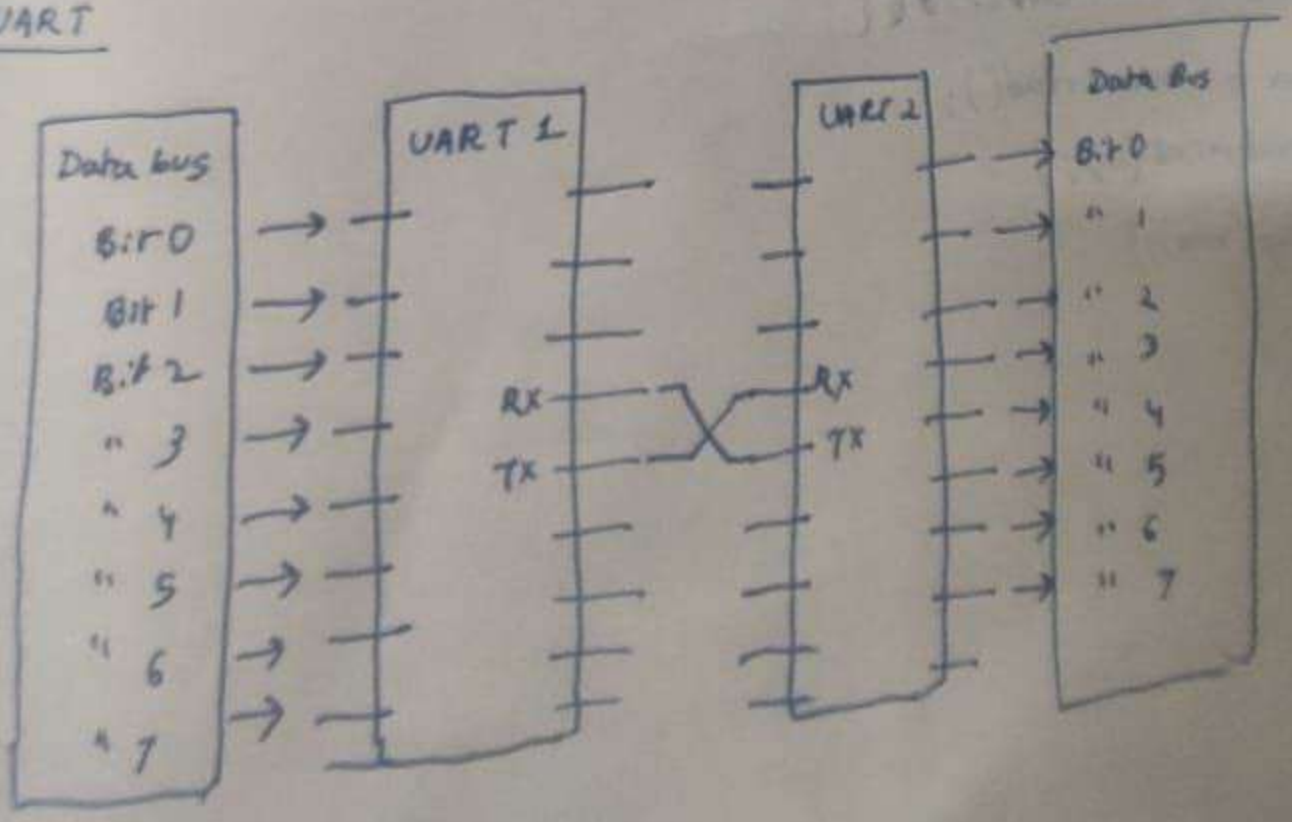


I2C



~~Wire Interface~~

UART



Wire Library

Its what Arduino uses to communicate with I2C devices.

Functions:

begin() - initialize I2C bus

end() - close I2C bus

requestFrom() - request bytes from peripheral devices

beginTransmission() - begins queueing up for transmission

endTransmission() - end the transmission

onReceive() - register a funcn to be called when a peripheral device receives [illegible]

onRequest() - " " " " " when a controller requests data.

Code for requesting 6 bytes of data from peripheral dev #8

```
#include <Wire.h>
void setup(){
 Wire.begin();
 Serial.begin(9600); }

void loop() {
 Wire.requestFrom(8,6);
 while(Wire.available()){
 char c = Wire.read();
 Serial.print(c); }
 delay(500);}
```

Temp converter with LCD

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 7, 8);

void setup(){
  lcd.begin(16, 2);
    Serial.begin(9600);

void loop(){ float celsius, farenheit;
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Enter Temp(C): ");
while(!Serial.available());
celsius = Serial.parseFloat();
farenheit = (celsius * 9.0 / 5.0) + 32.0;
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Temp(C): ");
lcd.print(celsius);
lcd.setCursor(0,1);
lcd.print("Temp(F): ");
lcd.print(farenheit);
delay(5000);
}
```

LOOP

Repeat a block of code multiple times.

for & while LOOP.

for LOOP

(Syntax)

```
for (initialization; condition; increment){
  }
```

while Loop

```
while (condition){
}
```

Eg for for,

Printing numbers from 1 to 5

```
for (int i=1; i<=5; i++){
  Serial.println(i);
  delay(1000);
}
```

Unit 4

Analog to Digital Conversion (ADC)

process of transforming a analog signals to digital form suitable for processing by computers.

Steps:

1) Sampling – capturing discrete samples of the analog signals at regular intervals
2) Quantization – mapping each sample's amplitude to the closest digital value.
3) Encoding – representing the quantized value in binary format.

ADC architectures

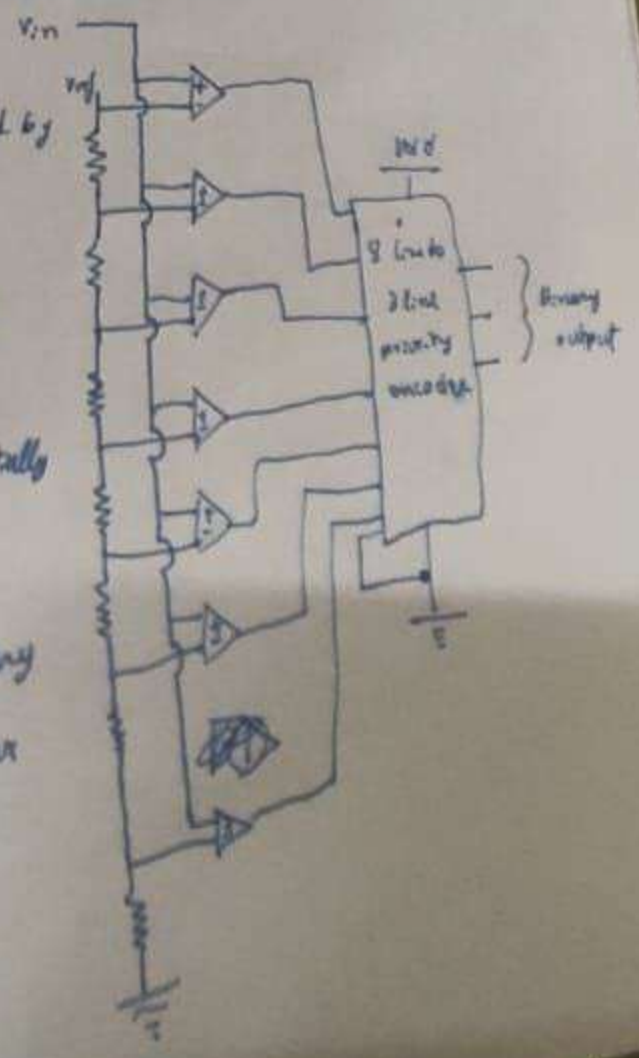
1) Successive Approximation ADC – iterative process to approximate the analog value.
2) Flash ADC – parallel comparison of input to reference voltages.
3) Sigma delta ADC – oversampling technique for high res & noise reduction.
4) Pipeline ADC – divides the conversion process into stages for higher speed.

Flash ADC

High speed & parallel ADC architecture.

Uses set of comparators to quickly determine digital output.

Each comparator compares signal to unique reference voltage.



Working:

V_{ref} is the stable ref. voltage provided by a voltage regulator.

As the analog input voltage exceeds ref. voltage at each comparator, the comparator will sequentially saturate to a high state.

The priority encoder generates a binary number based on the highest order active input, ignoring all other active inputs.

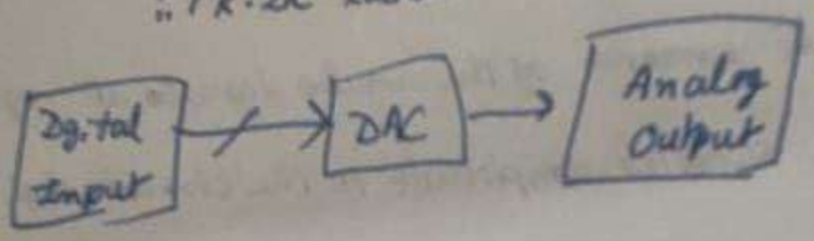
Digital to Analog Converter (DAC)

Converts digital input signal to analog output signal.

No of binary inputs is a power of 2.

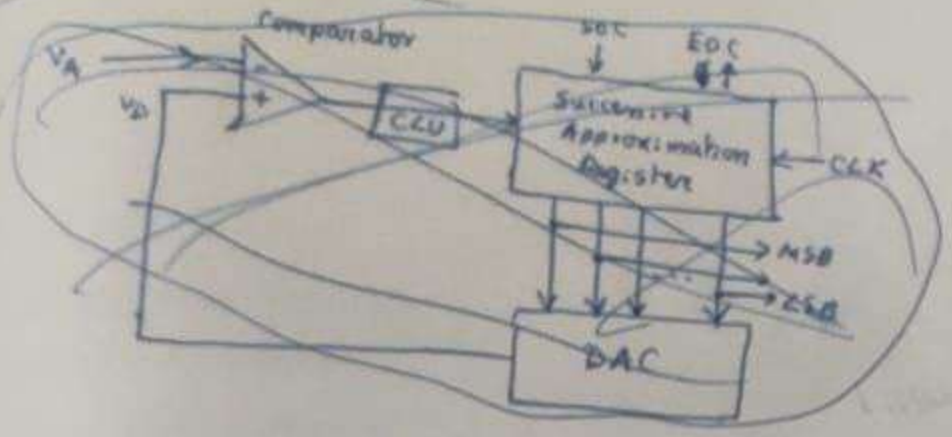
Types (2):- :> Weighted Resistor DAC

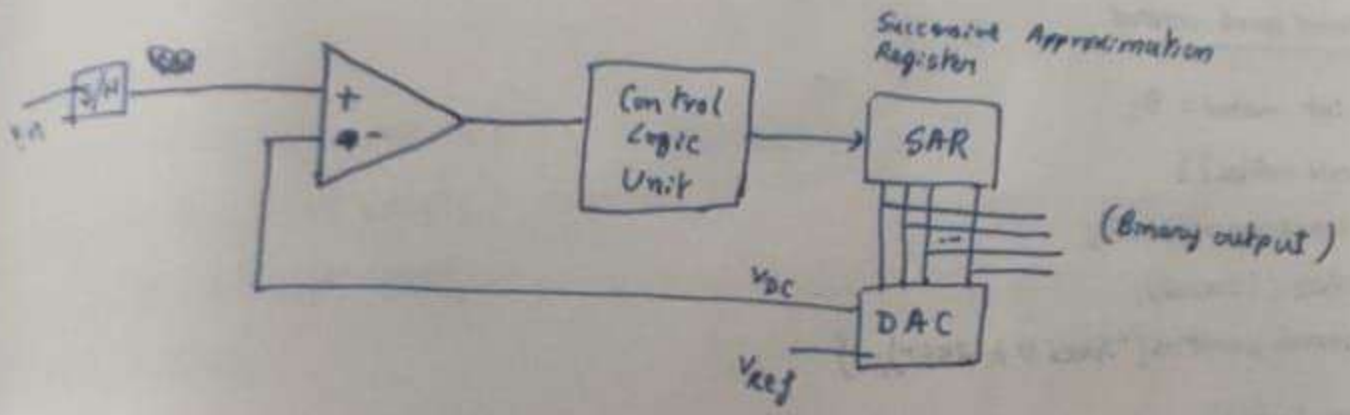
:> R-2R ladder DAC



3.6.1 R-2R Ladder DAC

Successive Approximation ADC





(4 - bit)

This consists of a comparator, a DAC & a SAR along with control logic unit. whenever, a new conversion starts, the sample and hold circuit samples the input signal. It is then compared with specific output signal of ~~ADC~~ DAC.

Lets say V_{in} is 5.6V & V_{ref} is 10V. When conversion starts, the SAR sets the most significant bit to 1 and others to 0. So, it becomes 1000, which for a 10V reference, the DAC will produce a value of 5V which is $\frac{V_{ref}}{2}$.

Now this will be compared with V_{in} and based on output, the output of SAR will change.

DC Motor

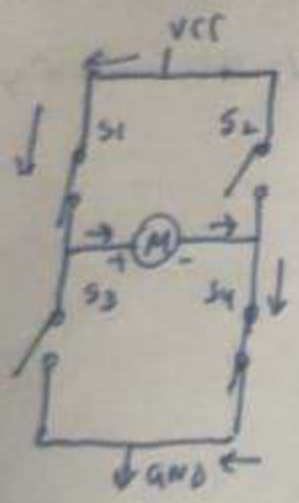
Motor spin

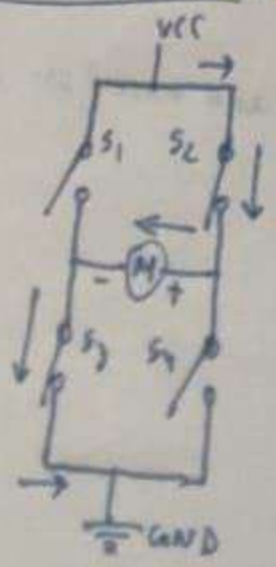
```
int motor = 9;
void setup (){
  }
void loop() {
digitalWrite(motor, HIGH); }
```

Motor speed control

```
int motor = 9;
void setup() {
Serial.begin(9600);
while (!Serial);
Serial.println("Speed 0 to 255"); }
void loop(){
if (Serial.available()) {
  int speed = Serial.parseInt();
  if (speed >=0 && speed <=255){
  analogWrite(motor, speed); }
}
}
```

H-Bridge Driver





There are 4 switches. S1 - S4.

When S1 & S4 are closed & S2 & S3 are open current flows in clockwise from VCC to GND

To reverse polarity of motor, S1 & S4 are ~~closed~~ open & now S2 & S3 are closed then the current flows in acw from VCC to GND and direction of motor rotation gets reverse

Motor Direction Control

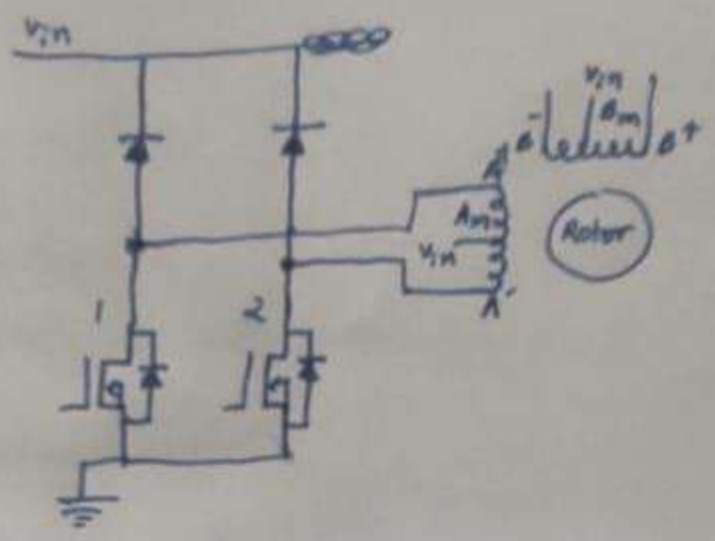
```
int pwm = 2;
int in1 = 8;
int in2 = 9;

void setup (){
  pinMode (pwm, OUTPUT);
  pinMode (in1, OUTPUT);
  pinMode (in2, OUTPUT); }

void loop(){
digitalWrite (in1, HIGH);          (for clockwise)
digitalWrite (in2, LOW);
analogWrite(pwm, 255);
delay (3000);

digitalWrite (in1, HIGH);          (Brake or motor stop)
digitalWrite (in2, HIGH);
delay (1000);
digitalWrite(in1, LOW);            (ACW)
digitalWrite (in2, HIGH);
delay (3000); }
```

Unipolar Stepper Motor



Bipolar Stepper Motor