Embedded Unit 5

⇒ Input / Output Programming:

(A) Digital Output Circuitry:

① To gain insight into how to interface controllers to external devices.

② Useful to examine the underlying circuitry.

③ Anything that can be controlled with a simple logical value, true/false input can be a target.

④ I/O port consists of 8 bits.

Arduino UNO Pin definitions:

→ pinMode(8, OUTPUT) : pinMode configures the specified Pin to behave as either Input or Output.

→ digitalWrite(): Write a HIGH or a LOW value to a digital pin.

Note: Digital Pin 13 is harder to use as a digital Input than the other digital pins because it has LED and Push up resistor attached to it.

Syntax: digitalWrite(pin, value). where pin = Pin number
value = HIGH or LOW.

→ delay() : Pauses the program for an amount of time specified as a parameter. ① The unit of delay is milliseconds.

Syntax: delay(ms)

→ digitalRead(): Reads a value from a specified digital Pin, either HIGH or LOW.

Syntax: digitalRead(pin) (Returns HIGH or LOW)

Example:

```
int ledPin = 13;
int inPin = 7;
int val = 0;
void setup()
{
pinMode(ledPin, OUTPUT);
pinMode(inPin, INPUT);
}
void loop()
{
val = digitalRead(inPin);
digitalWrite(ledPin, val);
}
```

⇒ GPS interfacing with Arduino:

GPS → Global Positioning System.

① Satellite based navigation system made up of atleast 24 satellites.

② Works in any whether conditions, anywhere in the world, 24 hours a day with no subscription fees or setup charges.

Working:

① GPS satellites circle the Earth twice a day in an orbit.

② Each satellite transmits a unique signal that allows GPS to decode and compute the precise location of the satellite.

③ GPS receivers use this information to calculate a user's exact location.

④ GPS receivers calculate the distance. With distance measurements, the receiver can determine a user's position and display it.

⑤ for 2D position → atleast 3 satellites. 3D → 4 or more satellites.

⑥ Once position has been determined, the GPS unit can calculate other information such as speed, trip distance, distance to destination, etc.

⑦ GPS signals will pass through clouds, glass and plastic but will not go through most solid objects, such as mountains & buildings. However, modern receivers are more sensitive and can usually track through houses.

GPS signal consists 3 different types of information.

① PseudoRandom Code: Identifies which satellite is transmitting info.
② Ephemeris data: Needed to determine a satellite's position.
③ Almanac data: Tells GPS receiver where each GPS satellite should be at any time.

→ Download and install required libraries for GPS to work:
(a) SoftwareSerial library
(b) TinyGPS library.

→ NEO-6M GPS MODULE

Overview:

(a) Neo-6M GPS chip: The heart of the module.
→ It can track upto 22 satellites.
→ Provides Power Save Mode (PSM). Making it suitable for power sensitive applications like GPS wristwatch.

(b) Position fix LED Indicator: → Blinks at various rates depending on what state it is in.
No blink → searching for satellites.
Blinks every 1s → Position fix is found.

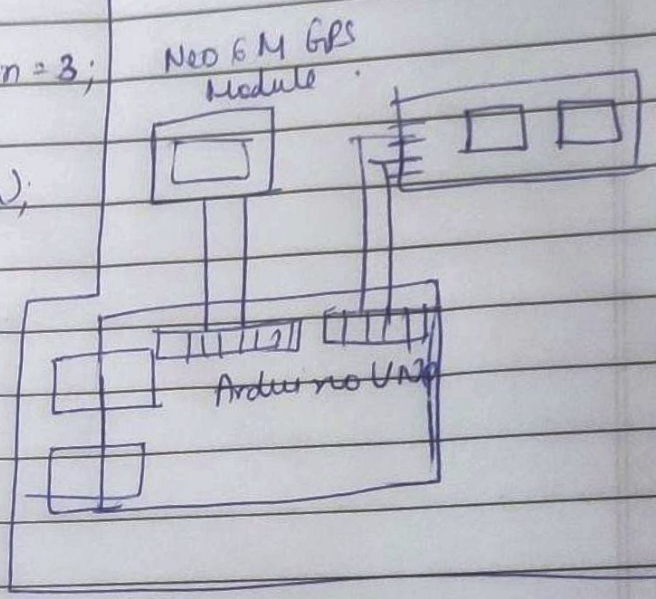
(c) Battery & EEPROM: 4KB in size. Connected to the chip via I2C.

→ Without the battery initial GPS takes more time
→ Battery is automatically charged when power is applied and maintains data for upto two weeks without power

DATE / /
PAGE

GPS.

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
static const int RXPin = 4, TXPin = 3;
TinyGPSPlus gps;
SoftwareSerial ss (RXPin, TXPin);
void setup() {
Serial.begin(9600);
ss.begin(9600);
}
void loop() {
while (ss.available()>0) {
    gps.encode (ss.read());
if (gps.location.isUpdated()) {
Serial.println("latitude = ");
Serial.println (gps.location.lat(), 6);
Serial.println("Longitude = ");
Serial.println (gps.location.lng(), 6);
}}}
```



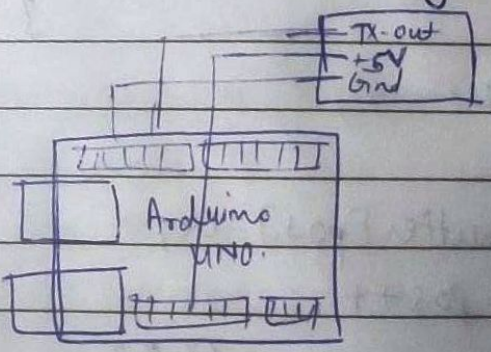
(*) Blood Pressure Sensor Interfacing:

→ The device which measures Blood Pressure is called Blood Pressure Sensor. It is also called Sphygmomanometer.

→ Features of Blood Pressure Sensor:

(a). Automatic Compression / Decompression.
(b) Easy to operate using buttons.
(c). Memory to save measurements.
(d). Automated Power Saving.
(e). Large LCD screen.
(f). Requires +5V for its operation.

following is the code to interface. The 3 parameters, systolic, diastolic and pulse Rate are separated by comma and space.




```
#include <SoftwareSerial.h>
int sensorValue;
SoftwareSerial serial(9,10);
char sbuffer[14], ch;
unsigned char pos;
unsigned char read1, read2, read3;
boolean newData = false;
void setup() {
Serial.begin(9600);
```

```
Serial.begin(9600);
Serial.println("Arduino is ready");
        }
void loop() {
recvChar();
showNewData();
        }
void recvChar(){
if(Serial.available()>0)
{
while(Serial.available()>0)
{
ch = Serial.read();
if(ch==0x0A)
{
pos=0;
newData=true;
read1=((sbuffer[1]-'0')*100)+((sbuffer[2]-'0')*10)+(sbuffer[3]-'0');
read2=((sbuffer[6]-'0')*100)+((sbuffer[7]-'0'*10)+(sbuffer[8]-'0');
read3=((sbuffer[11]-'0')*100)+((sbuffer[12]-'0'*10)+(sbuffer[13]-'0');
        }
else
{
sbuffer[pos]=ch;
  pos++
        }}}}
void showNewData(){
if(newData==true)
   {
    Serial.println("Calculating Results....");
    Serial.println(read1);
```

```
Serial.println(read2);
Serial.println(read3);
NewData = false;
    }
}
```

(*) Bluetooth Interfacing:

→ HC 05/06 works on serial communication.
→ The Bluegiga WT11 module on Arduino BT provides Bluetooth communication with computers, phones and other bluetooth devices.
→ Bluetooth modules can transmit and receive data wirelessly by using two devices.
→ HC 05 is used for wireless communication. It communicates with microcontrollers using serial communication. (USART)
→ Default settings of HC-05 bluetooth module can be changed using certain AT commands.
→ Bluetooth is used to transfer data b/w electronic devices over short distances.
→ Unlike wifi, it does not need a password to connect.
→ Connection is usually not secure as wifi.
→ Bluetooth is slower than wifi and has shorter range.
→ Replace cables connecting electronic devices.

```
#include<SoftwareSerial.h>
SoftwareSerial bt(2,3);
void setup()
{
bt.begin(9600);
Serial.begin(9600);
    }
void loop()
{
if(bt.available())
{
Serial.write(bt.read());
                    }}
```

* HeartBeat Sensor Using Arduino (Heart Rate Monitor)

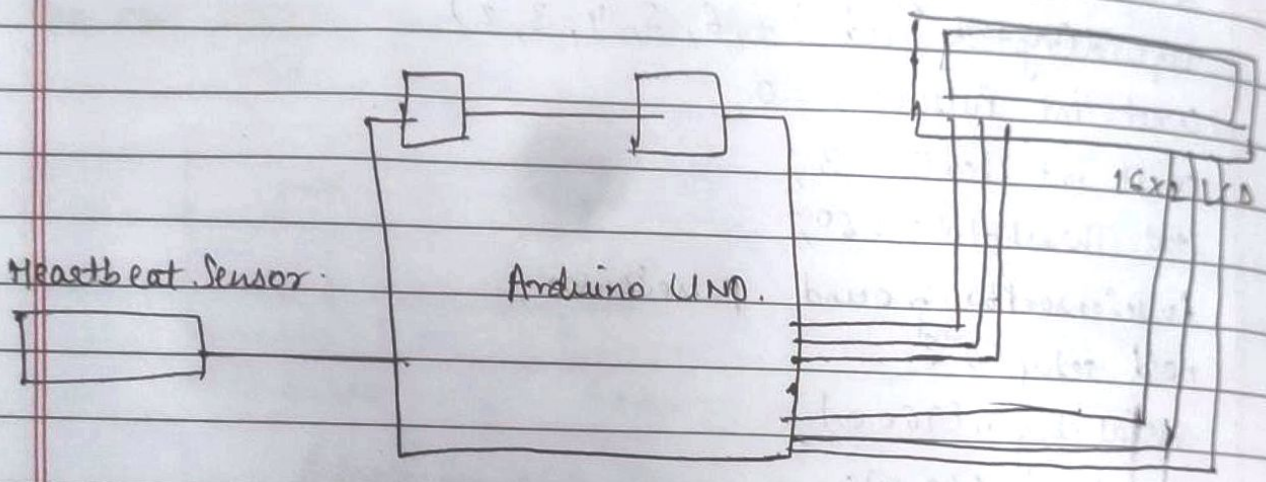
→ An electronic device used to measure Heart Rate, i.e., speed of Heartbeat.
→ Heart Rate can be monitored in 2 ways:
① Manually check the pulse either at wrist or neck.
② Use a Heartbeat sensor.
→ There are many ways to measure Heart Rate and the most precise one is using Electrocardiography.
→ More easy way is to use a HeartBeat sensor.
→ Indicates the no. of times the heart is contracting or expanding in a minute.
→ Heartbeat sensors are available in Wristwatches (Smart watches), Smart Phones, chest straps, etc.
→ Principle behind the working of Heartbeat Sensor: The changes in the volume of the blood in an organ is measured by the changes in the intensity of the light passing through that organ. (Photoplethysmograph)

→ Usually, the source of light would be IR LED and detector would be any photo detector like a Photo Diode or LDR (Light Dependent Resistor) or a Photo Transistor.

→ These can be arranged in two ways:

(a) Transmissive Sensor: light source and detector are placed facing each other.

(b) Reflective Sensor: light source and detector are placed adjacent to each other.



Working:

→ Upload the code.

→ Arduino asks to place our finger in the sensor and press the switch.

→ Place any finger (except the thumb) and press the switch.

→ Based on the data from sensor, Arduino calculates the heart rate and displays on LCD.

→ While the sensor is collecting data, sit down and relax and do not shake the wire as it might result in faulty values.

→ After result is displayed if you want to perform another test, just push the rest button and start the procedure once again.

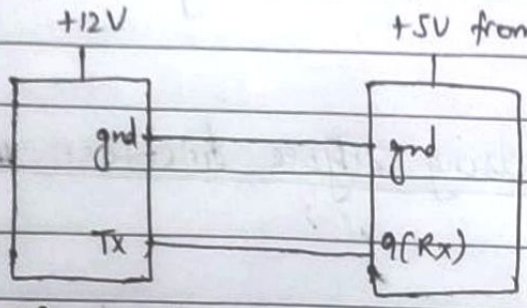
Code:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(6, 5, 3, 2, 1, 0);
```

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2)
const int PulseWire = 0;
const int LED13 = 13;
int Threshold = 550;
PulseSensorPlayground pulseSensor;
void setup() {
Serial.begin(9600)
lcd.begin(20, 4);
pulseSensor.analogInput(PulseWire);
pulseSensor.blinkOnPulse(LED13);
pulseSensor.setThreshold(Threshold);
if (pulseSensor.begin()) {
lcd.setCursor(0, 0);
lcd.print("Heart Rate Monitor");
}}
void loop() {
int myBPM = pulseSensor.getBeatsPerMinute();
if (pulseSensor.sawStartOfBeat()) {
Serial.println("A heartbeat happened");
Serial.println(myBPM);
lcd.setCursor(5, 3)
lcd.print(myBPM);
}  delay(20);  }
```

(*) Interfacing RFID with Arduino. - How to read RFID cards using Arduino.

→ RFID - Radio frequency Identification.
→ The chip inside RFID tag gets power via Electromagnetic Induction.
→ Both RFID reader and RFID tag comes with a coil in them.
→ When an RFID tag is shown near the reader, electromagnetic Induction will take place b/w coils and this powers the chip inside tag.
→ This chip will send data electromagnetically to reader.
→ We can collect the read data through serial output pins.



Interfacing RFID Reader to Arduino.

```
#include<SoftwareSerial.h>
SoftwareSerial myserial(9,10);
void setup()
{
myserial.begin(9600);
Serial.begin(9600);
}
void loop(){
if(myserial.available()>0)
{
Serial.write(myserial.read());
}
}
```

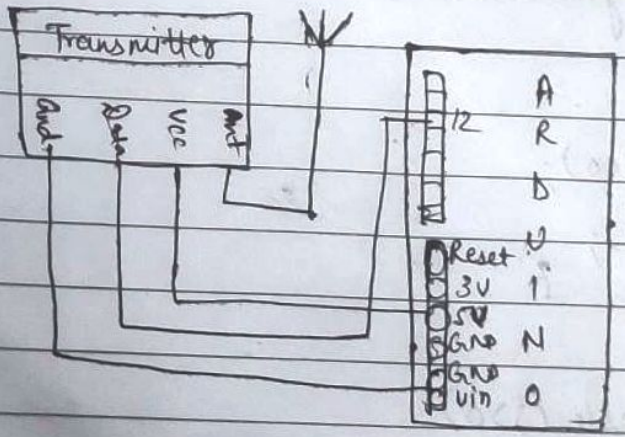
→ mySerial.available() → checks for any data coming from RFID reader module through software serial pin 9.
→ mySerial.read() → Reads the incoming data through software serial port.
→ Serial.write() → Prints data to serial monitor of Arduino.

→ Each RFID tag is a 12 character unique number.
→ We need a two dimensional array to store multiple RFID tags. To store 10 RFID cards, we need an array of 10 rows and 12 columns.

(*). Wireless Communication using ZigBee Interface with Arduino UNO

Wireless Communication:

→ To communicate with Arduino from a long distance without wires.
→ Low cost is primary requirement.
→ A number of different Xbee modules are available.



Simple Wireless transmitter Using Virtual Wire.

On the transmitting side, the code will be:

```
#include <SoftwareSerial.h>
SoftwareSerial xbeeSerial(2,3);
void setup(){
Serial.begin(9600);
xbeeSerial.begin(9600);
                }
void loop(){
 if(Serial.available()>0){
  char input = Serial.read();
  xbeeSerial.print(input);
                    }}
```

The code for receiving side is:

```
#include <Softwareserial.h>
SoftwareSerial xbeeSerial(2,3);
void setup(){
Serial.begin(9600);
xbee Serial.begin(9600);
                }
void loop(){
   if(xbee Serial.available()>0){
          char input = xbee Serial.read();
          Serial.print(input);
                        }}
```