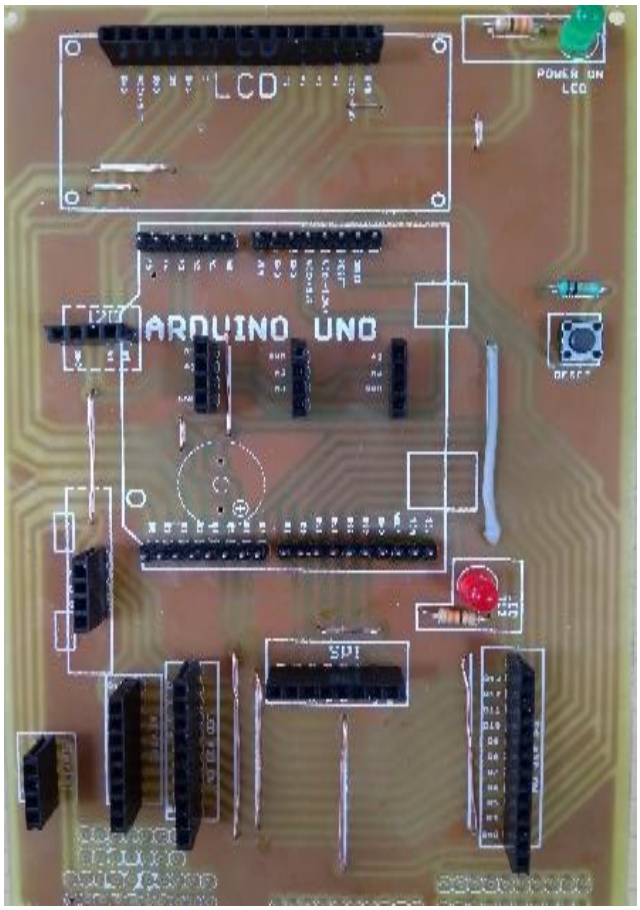
Arduino Kit

Chandigarh



Brief Content List

Figure Index

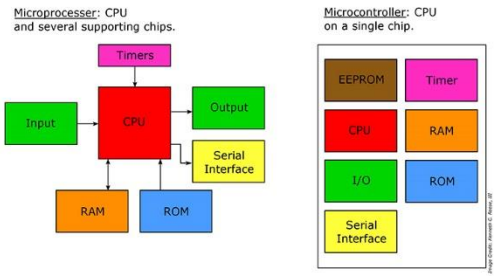
CODE SNIPPET INDEX

INTRODUCTION

In today's world basic electronics is part of every school curriculum. It is indeed the need of the hour, as every vertical of life nowadays involves some or the other form of electronic devices to make the accessibility of recourses easier in the modern world.

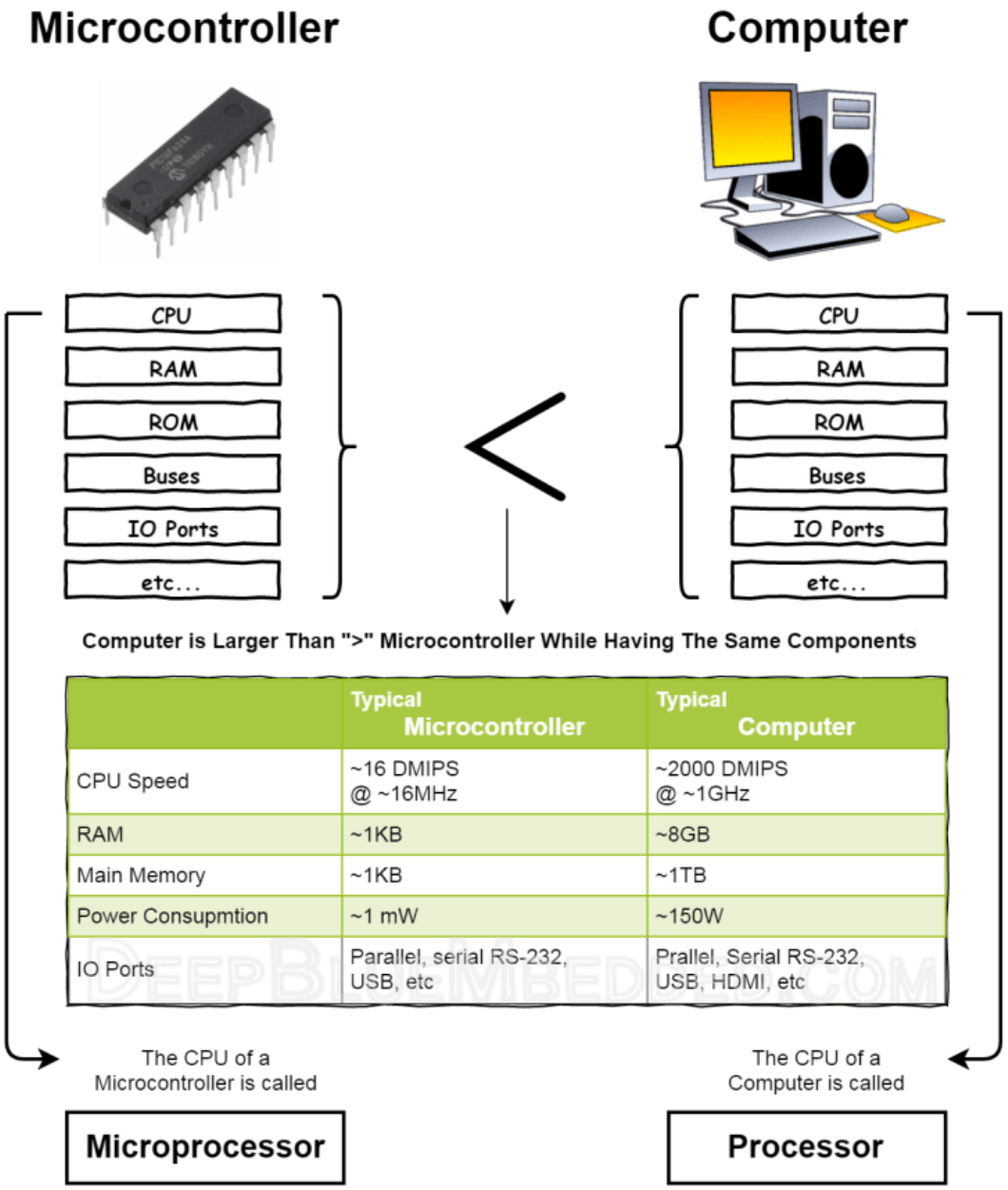
Microcontroller is the essential component of every electronic devices being used in this modern connected world. Microcontroller is analogous to the small compressed computer fabricated on single chip, which controlled all the embedded devices.

> "THE INTERNET OF THINGS (IOT) IS THE NETWORK OF PHYSICAL OBJECTS THAT CONTAIN EMBEDDED TECHNOLOGY TO COMMUNICATE AND SENSE OR INTERACT WITH THEIR INTERNAL STATES OR THE EXTERNAL ENVIRONMENT."
> GARTNER ..



Microprocessor is essential component of any modern PC.

Microprocessor	Microcontroller
• CPU is separate from RAM, ROM, I/O, timer, etc • User can choose the amount and specification of RAM, ROM, I/O ports	• CPU, RAM, ROM, I/O, timer all are on single chip • Fixed amount of on-chip RAM, ROM, I/O ports
• Costly • General purpose • High processing power	• Cheap • Specific/single purpose • Low processing power



Source : https://deepbluembedded.com/

Source : arduino.cc

What is 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. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is open-source, and it is growing through the contributions of users worldwide.

Why Arduino?

Thanks to its simple and accessible user experience, Arduino has been used in thousands of different projects and applications. The Arduino software is easy-to-use for beginners, yet flexible enough for advanced users. It runs on Mac, Windows, and Linux. Teachers and students use it to build low cost scientific instruments, to prove chemistry and physics principles, or to get started with programming and robotics. Designers and architects build interactive prototypes, musicians and artists use it for installations and to experiment with new musical instruments. Makers, of course, use it to build many of the projects exhibited at the Maker Faire, for example. Arduino is a key tool to learn new things. Anyone - children, hobbyists, artists, programmers - can start tinkering just following the step by step instructions of a kit, or sharing ideas online with other members of the Arduino community.

There are many other microcontrollers and microcontroller platforms available for physical computing. Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, and many others offer similar functionality. All of these tools take the messy details of microcontroller

programming and wrap it up in an easy-to-use package. Arduino also simplifies the process of working with microcontrollers, but it offers some advantage for teachers, students, and interested amateurs over other systems:

- Inexpensive - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- Cross-platform - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows.
- Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the Processing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- Open source and extensible software - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- Open source and extensible hardware - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

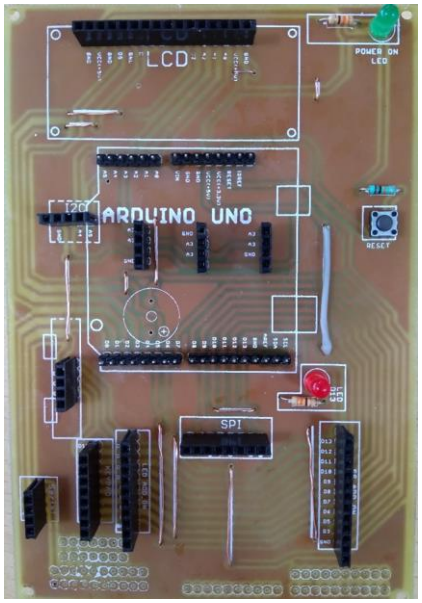
1. GETTING STARTED

Discuss Arduino IDE, open source writing first program , compiling , uploading, port…..

The Arduino project was conceived in Design school of Italy, to promote programming at school level and to promote inclination towards hardware interfacing in real world.

Arduinokit is designed to make school students understand the basics of programming and hardware interfacing at novice level. The step by step experiments planned using "*ArduinoKit*" helps the users to achieve better understanding in the areas of Embedded Electronics, which is the future of computing world. The pedagogy behind the entire curricula is *"to make the technological understanding simple"*

ArduinoKit



The *ArduinoKit* can be used to perform basic experiment to advance level of hardware interfacing. The experiments designed, makes user understand:

- digital input/output using LED, button (tac switch) interfacing.
- analog input/output or PWM
- interfacing hardware using protocols like SPI, I2C, Serial communication is also available.

The other words students can perform following experiments to augment their knowledge.

- Interfacing multiple LEDs, Button, LED fading
- LCD, Keypad interfacing
- Two seven segment interfacing
- Real time clock using I2C protocol
- Rfid reader using SPI protocol
- Ultrasonic sensor interfacing
- Analog sensor (MQ-7 / MQ-135)
- Ultrasonic sensor
- Interfacing Bluetooth –HC05 using serial communication
-

2. HARDWARE DESCRIPTION

Kit description, different add-on boards….

2.1. Block Diagram of Kit

The figure below shows the location of different components on the kit.

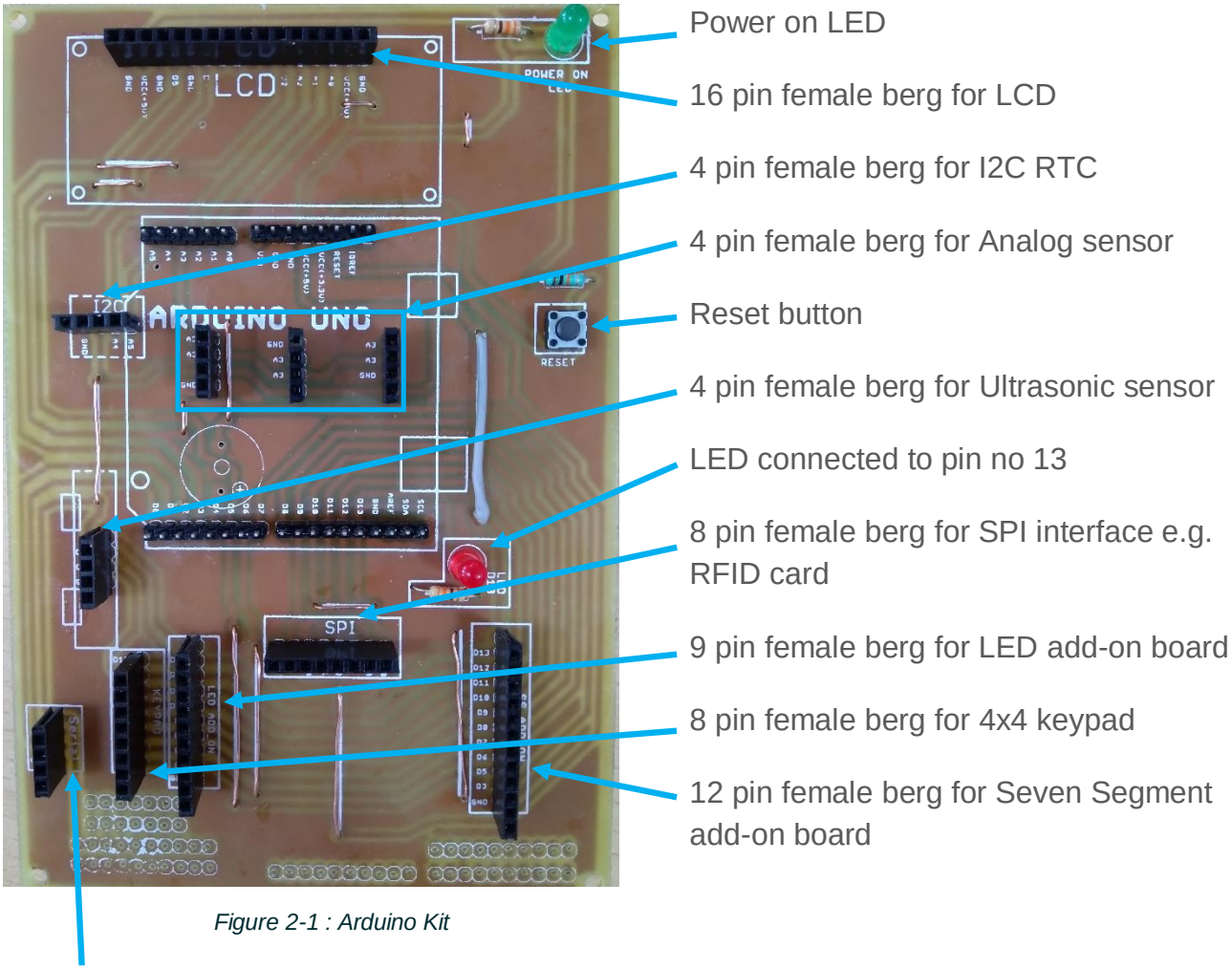


Figure 2-1 : Arduino Kit

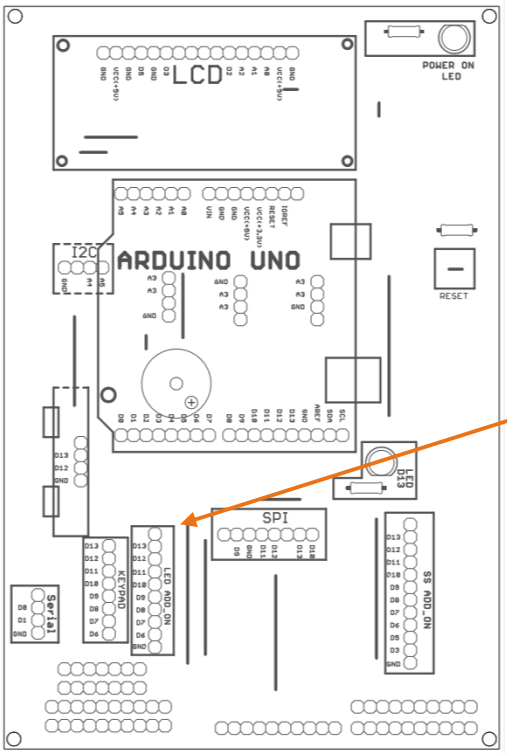
-

3. CONNECTOR DETAILS

Discuss all the pins in each connector, label, Vcc(+ve), Gnd, D1, A0, etc

The connectors

3.1. LED Add-on



LED Button ADD-ON

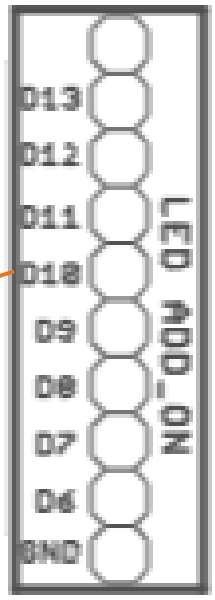


Figure 3-1 : LED Button ADD-ON

Pin Description

	+5v
D13	Button 1
D12	Button 2
D11	LED 1
D10	LED 4
D9	LED 2
D8	Button 3
D7	Button 4
D6	LED 3
	Gnd

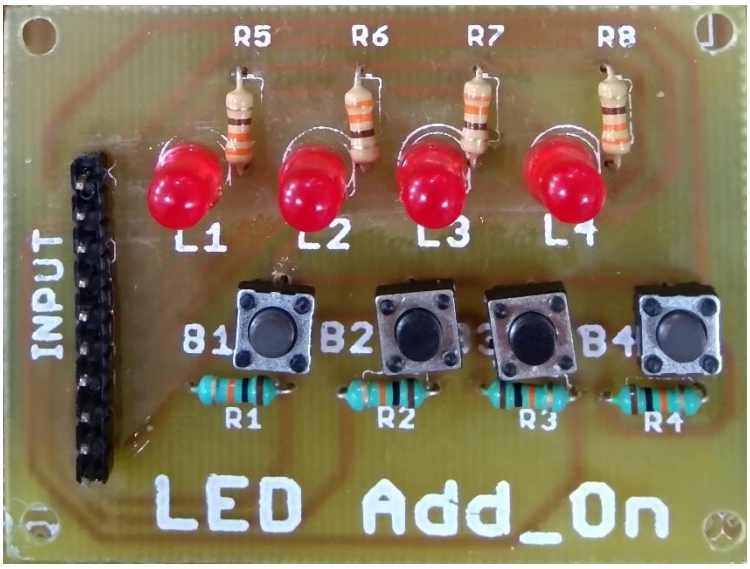


Figure 3-2 : LED Button ADD_ON PCB

R5, R6, R7 and R8 resistors are of 330Ω value and are connected as current limiting resistors, serial with LEDs L1, L2, L3 and L4 respectively.

R1, R2, R3 and R4 resistors are of 10KΩ rating. These are connected as pull down resistor with button B1, B2, B3 and B4 respectively.

3.2. Buzzer – Digital Output

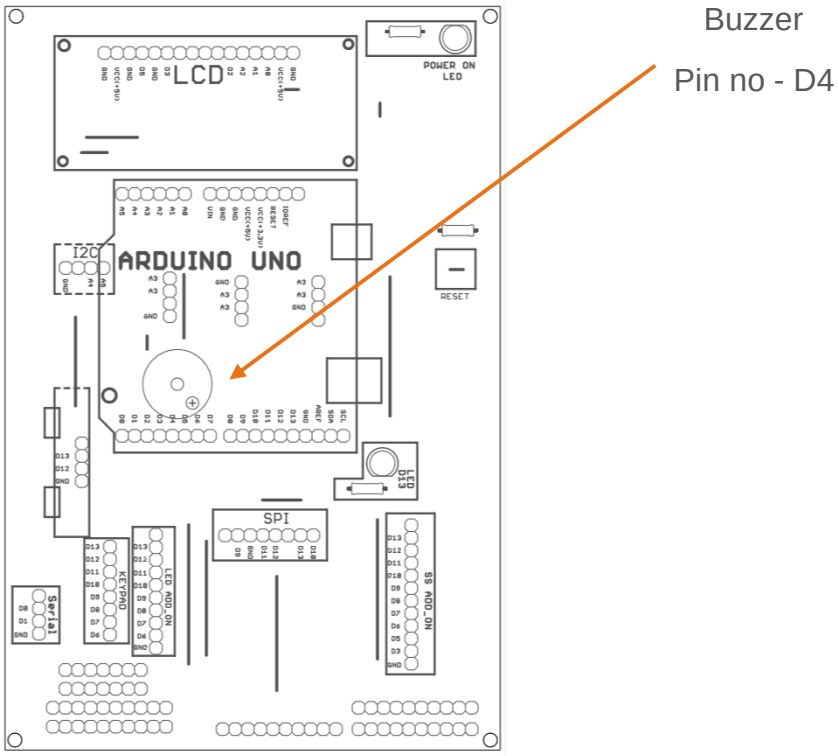
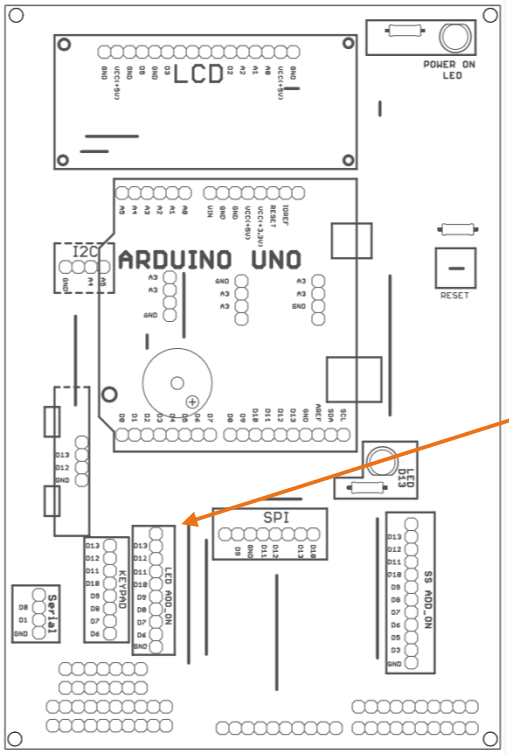


Figure 3-3 : Buzzer

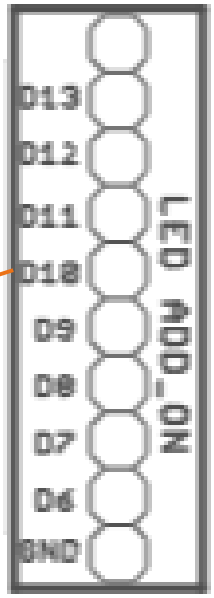
3.3. Analog Input

3.4. PWM – LED – Analog Output



LED Button ADD-ON

LED Button ADD-ON



Pin Description

	+5v
D13	Button 1
D12	Button 2
D11	LED 1 (PWM)
D10	LED 4 (PWM)
D9	LED 2 (PWM)
D8	Button 3
D7	Button 4
D6	LED 3 (PWM)
	Gnd

3.5. Liquid Crystal Display (LCD)

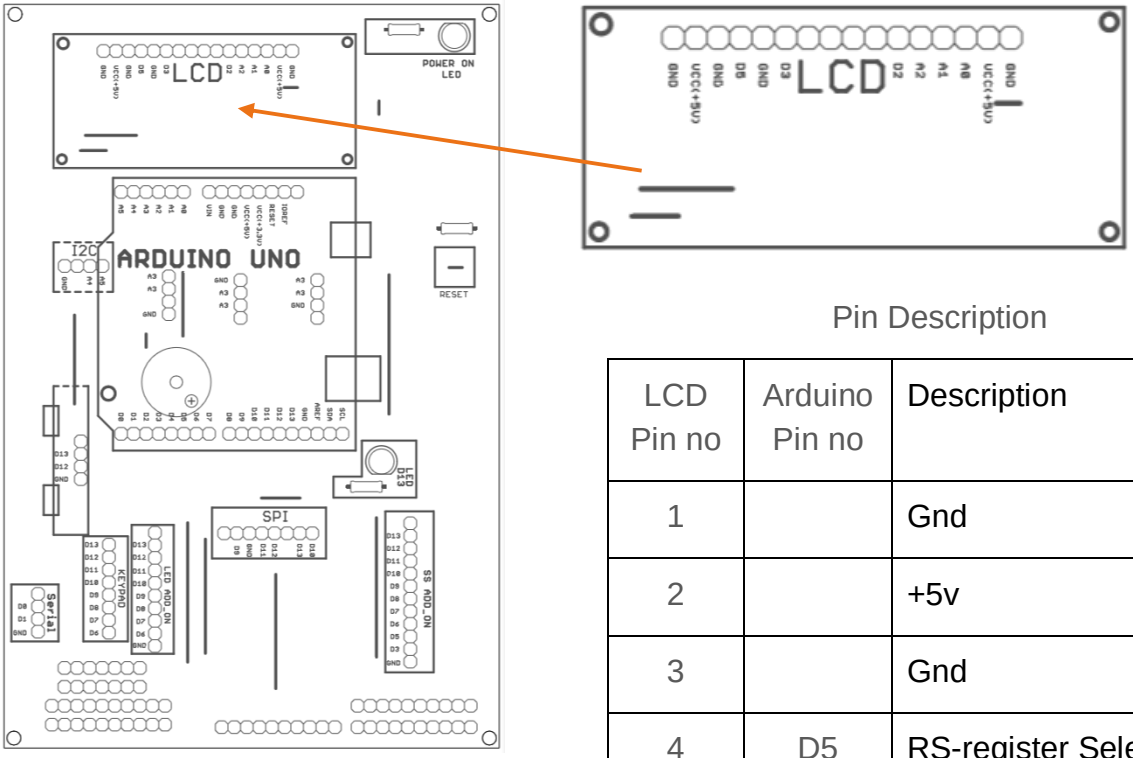
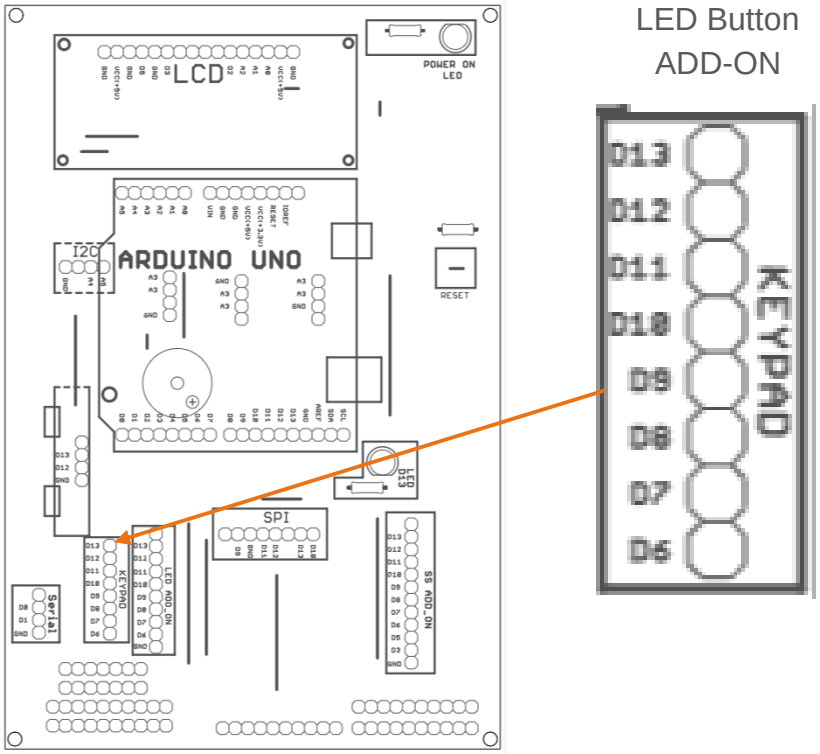


Figure 3-4 : LCD

Pin Description

LCD Pin no	Arduino Pin no	Description
1		Gnd
2		+5v
3		Gnd
4	D5	RS-register Select
5		Gnd
6	D3	E-enable
7-10	No connection	
11	D2	Data pin 4
12	A2	Data pin 5
13	A1	Data pin 6
14	A0	Data pin 7
15		+5v
16		Gnd

3.6. Keypad



Pin Description

D13	Row 1
D12	Row 2
D11	Row 3
D10	Row 4
D9	Column 1
D8	Column 2
D7	Column 3
D6	Column 4

Figure 3-5 : Keypad

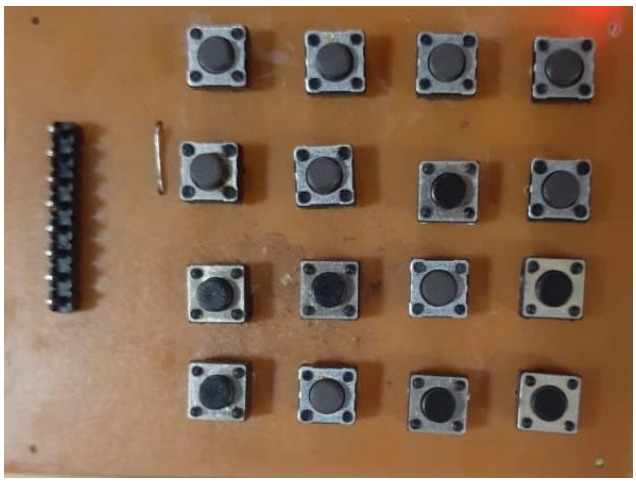


Figure 3-6 : Keypad Add-on

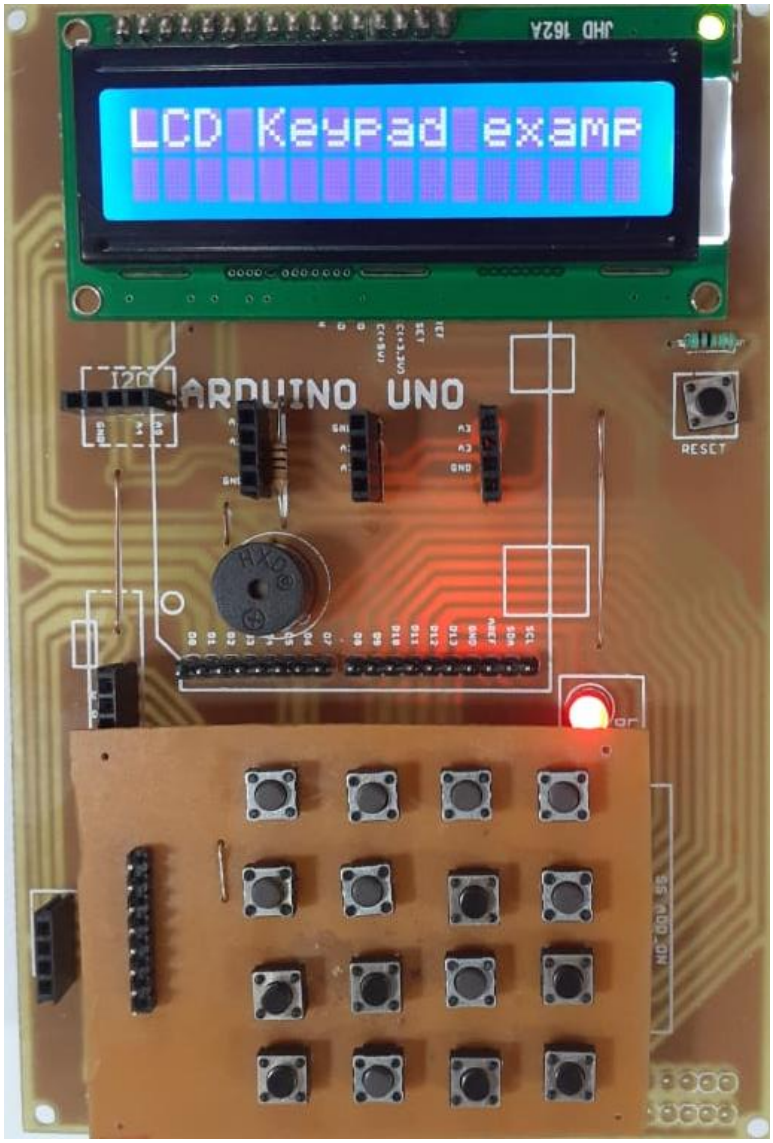


Figure 3-7 : Keypad LCD interfacing

3.7. Seven Segment

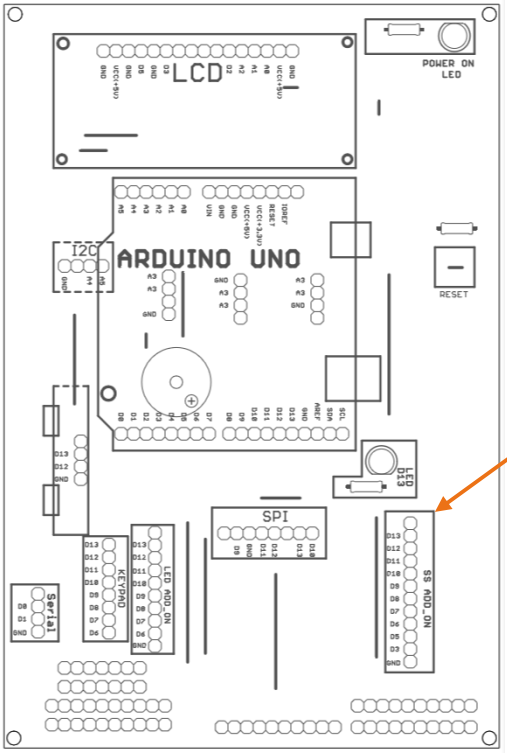
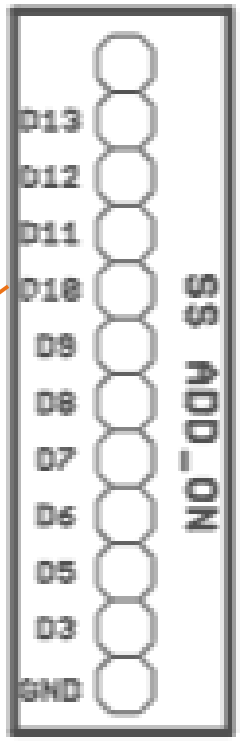


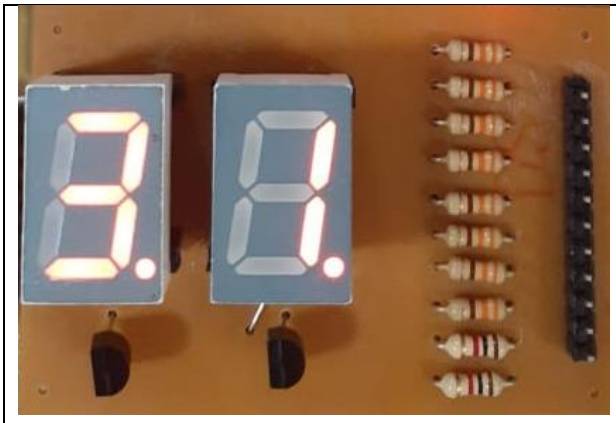
Figure 3-8 : Keypad

Seven Segment ADD-ON



Pin Description

D13	DP
D12	C
D11	D
D10	E
D9	G
D8	F
D7	A
D6	B
D5	Transistor 1
D3	Transistor 2
	Gnd

 *Figure 3-9 : Seven Segment Display Add-on*	Common anode seven segment display (SSD)

3.8. I2C – Real Time Clock (RTC)

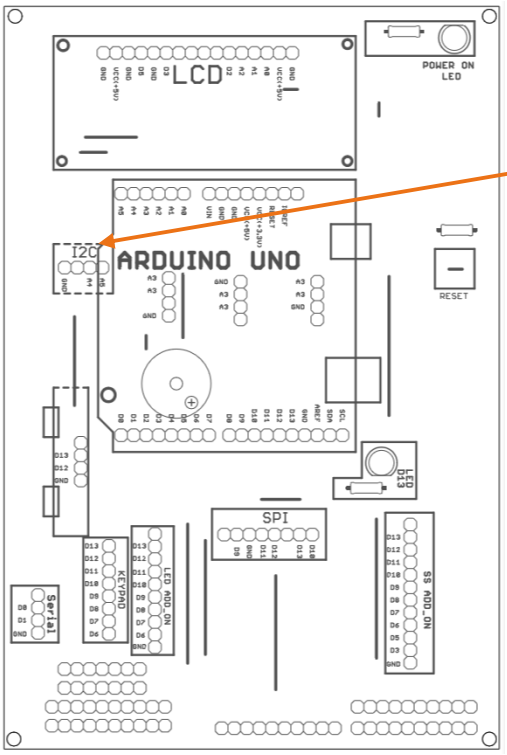
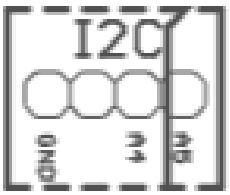
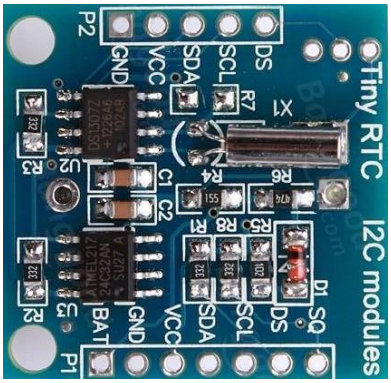


Figure 3-10 : SPI - RFID Add-on

I2C (RTC) ADD-ON





Pin Description

	Gnd
	+5V
A4	SDA
A5	SCL

3.9. SPI – RFID

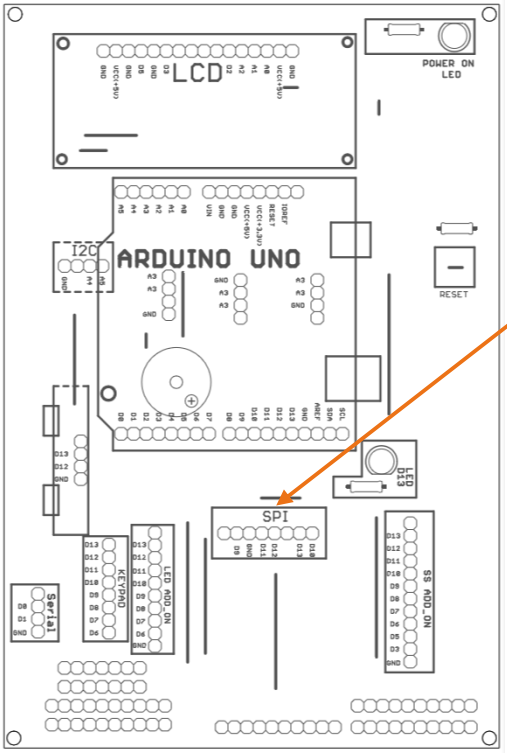
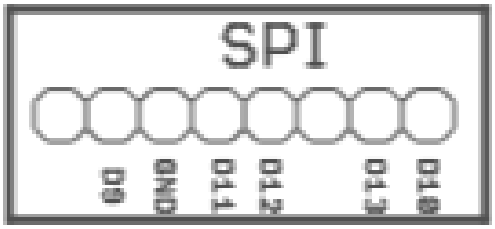


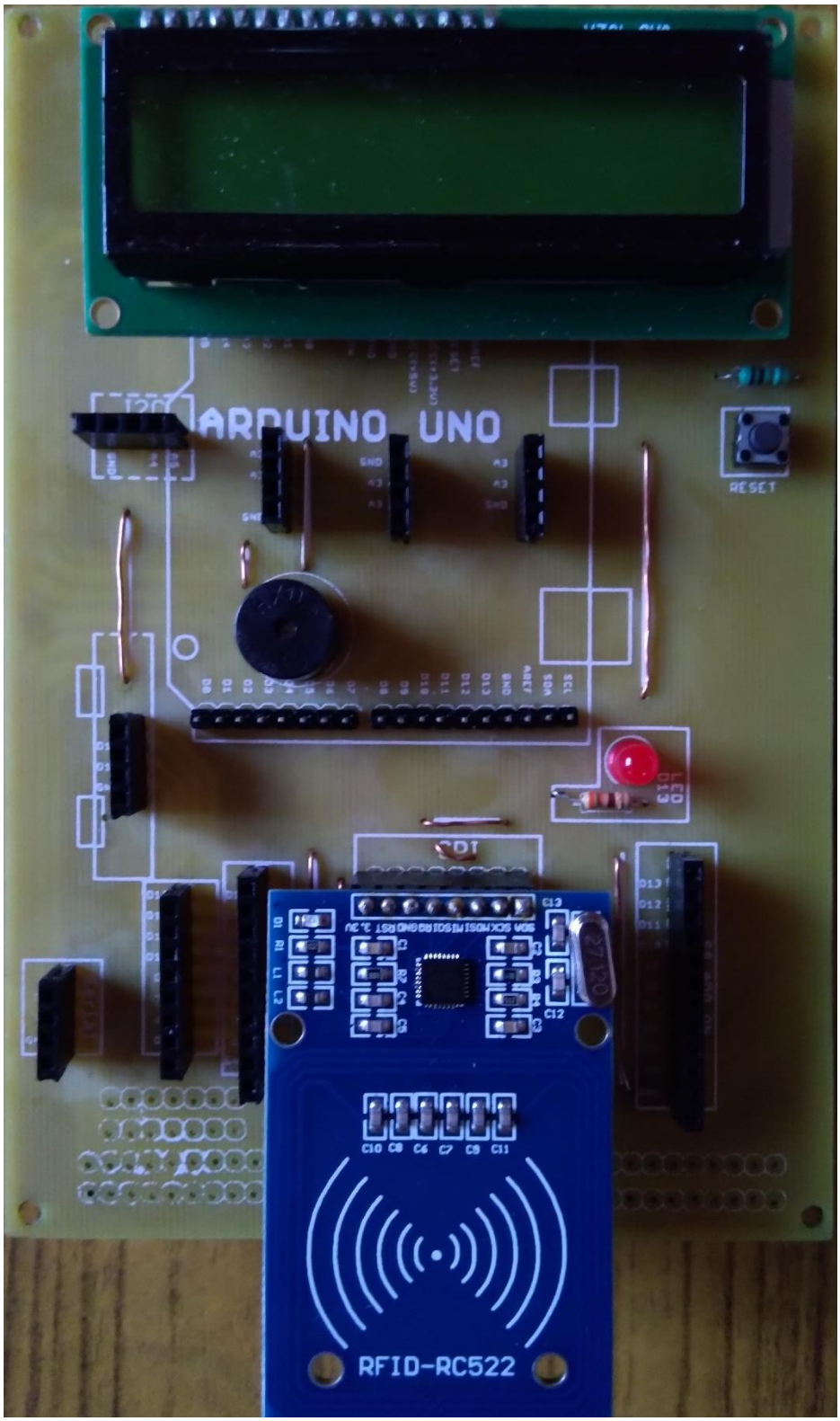
Figure 3-11 : SPI - RFID Add-on

SPI (RFID) ADD-ON



Pin Description

	3.3 v
D9	Reset
	Gnd
D11	RQ
D12	MISO
	MOSI
D13	SCK
D10	SDA


ARDUINO UNO
RESET
RFID-RC522

3.10.Ultrasonic Sensor

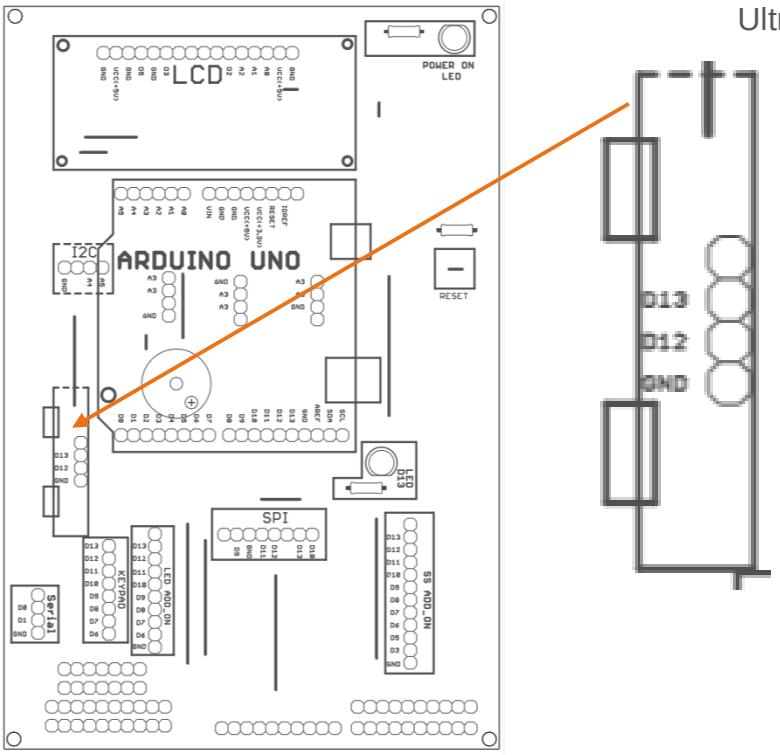


Figure 3-12 : Ultrasonic Sensor Add-on

Ultrasonic sensor (HC-SR04)

Figure 3-13 : HC-SR04

Pin Description

	+5V
D13	Trigger
D12	Echo
	Gnd

3.11.Analog Sensor - MQ-7, MQ135

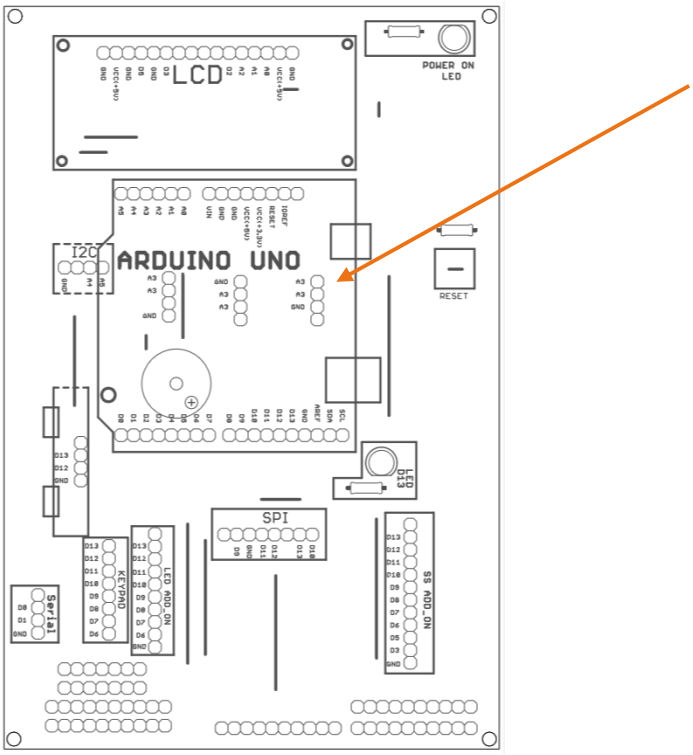
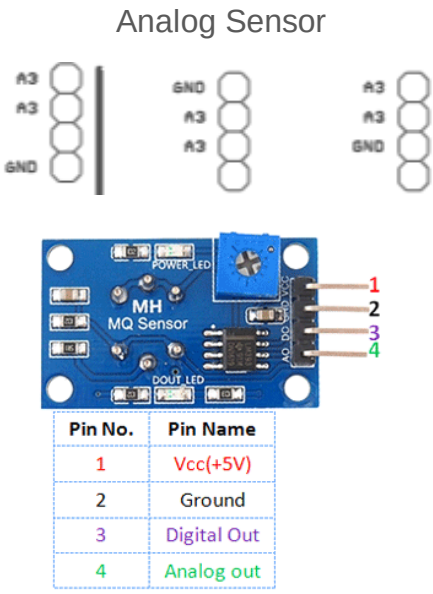


Figure 3-14 : Analog Sensors



Pin No.	Pin Name
1	Vcc(+5V)
2	Ground
3	Digital Out
4	Analog out

Figure 3-15 : MQ-7 sensor

3.12.Serial Communication – Bluetooth (HC-05)

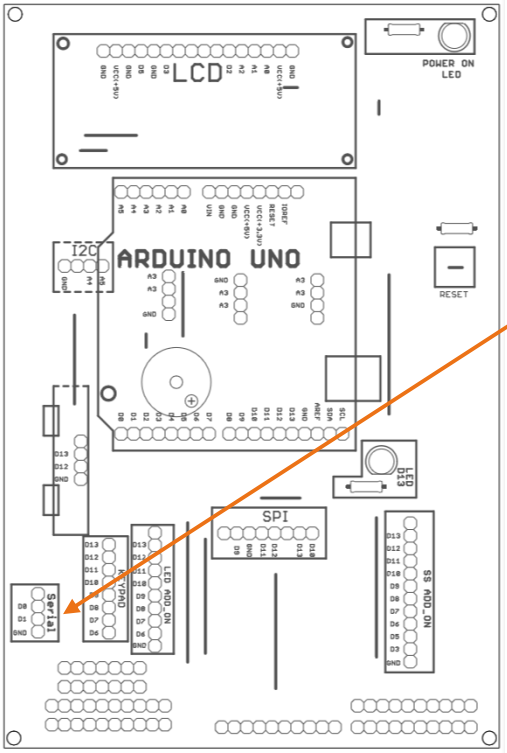
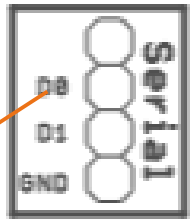


Figure 3-16 : Serial Communication



Pin Description

	+5v
D0	Rx
D1	Tx
	Gnd

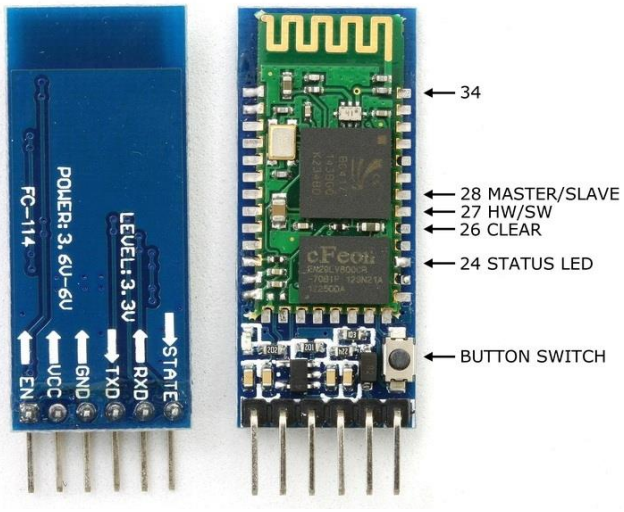
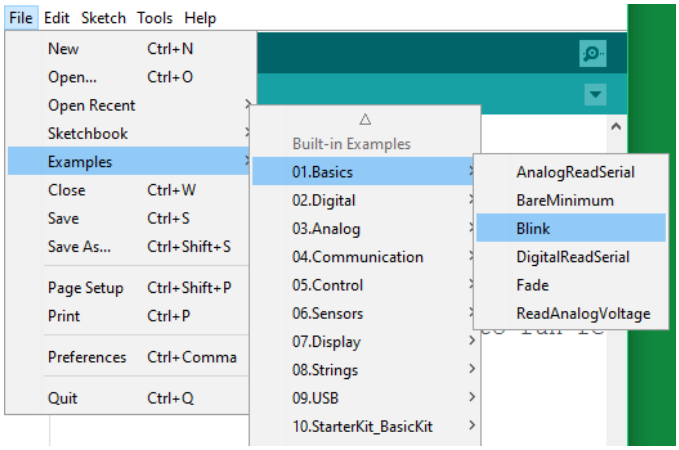


Figure 3-17 : HC-05 Bluetooth Module

3.13. I2C - OLED

4. EXPERIMENTS

5.1. LED Blinking



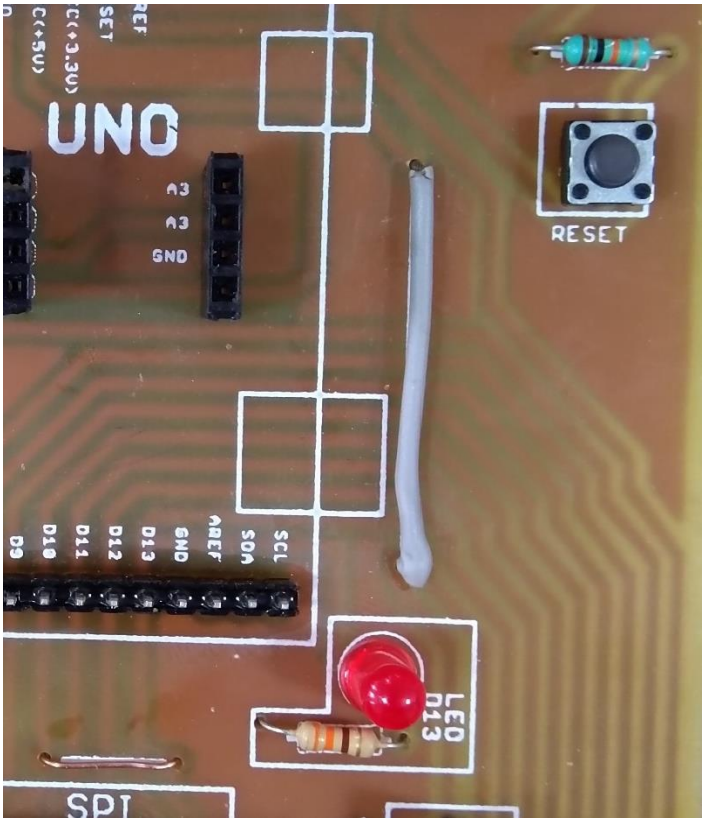
File > Examples > 01.Basics > Blink

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Code Snippet 1 : LED blinking

Once the code is compiled and uploaded in the Arduino Kit. The LED connected at pin no. 13 starts blinking on and off with delay of 1000milliseconds or 1 second.



5.2. Using ArduinoKit Library

5.3. Buzzer interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 2 : Buzzer

5.4. Button Interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 3 : Buzzer

5.5. Seven Segment interfacing

Use the Seven Segment Display (SSD) Add-on board

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 4 : Buzzer

5.6. LCD interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 5 : Buzzer

5.7. Keypad interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 6 : Buzzer

5.8. I2C - RTC interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 7 : Buzzer

5.9. SPI - RFID interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 8 : Buzzer

5.10. Ultrasonic Sensor interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 9 : Buzzer

5.11. Sensor interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 10 : Buzzer

5.12. Serial Communication - Bluetooth interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 11 : Buzzer

5.13. I2C - OLED interfacing

```
// Buzzer on – off at the frequency of 1 second.
void setup() {
  // initialize digital pin 4 as an output.
  pinMode(4, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(4, HIGH);   // turn the Buzzer on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(4, LOW);    // turn the Buzzer off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Code Snippet 12 : Buzzer

5. PROJECTS

Additional projects

5.1. Password based security system using LCD-Keypad
5.2.