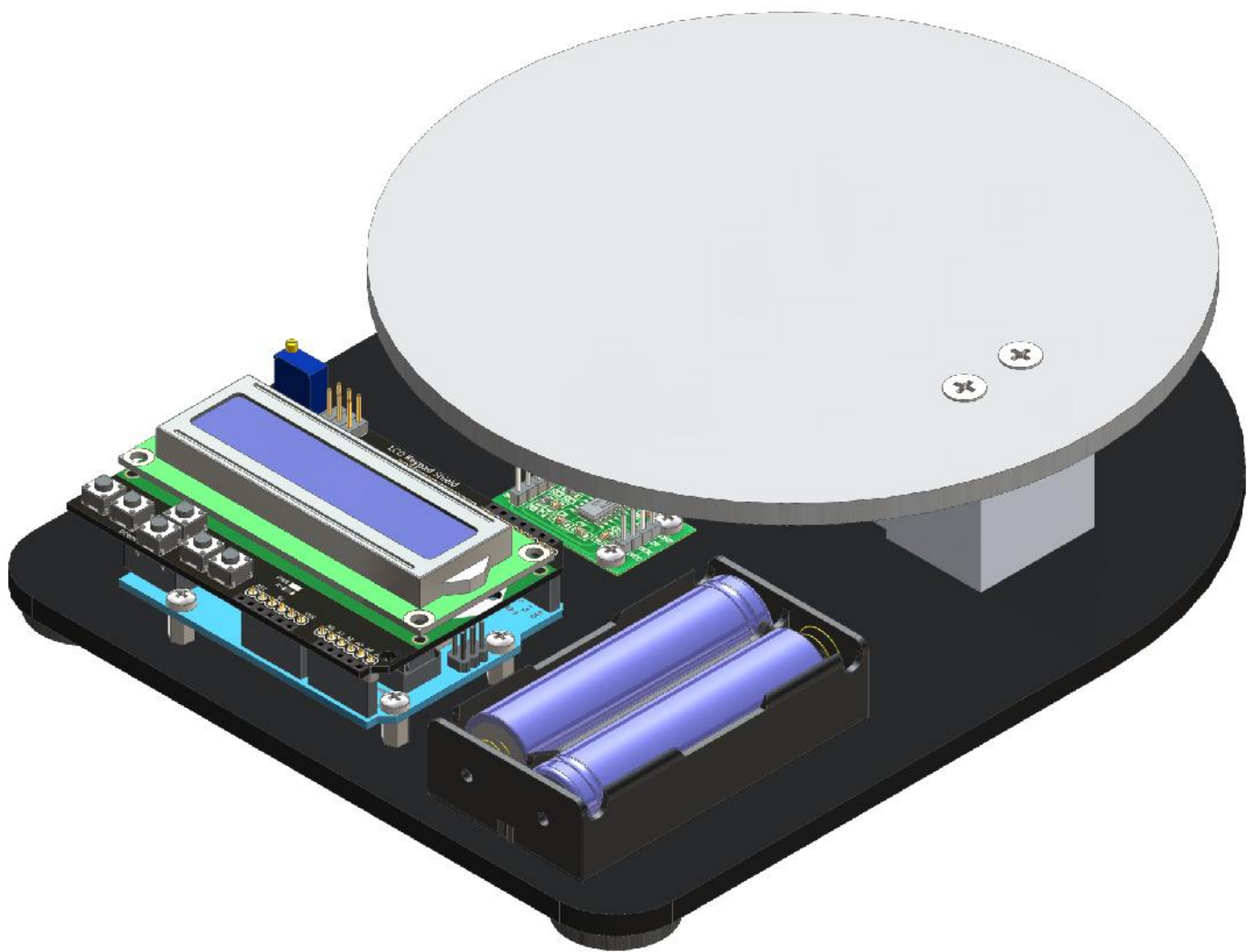


WEIGHING SCALE DIY EVALUATION KIT (20KG)



SunRobotics
Parts for Every Idea!

www.sunrobotics.in



SunRobotics Technologies

C-301, Sumel Business Park 6, Dudheshwar Road, Ahmedabad-380004, Gujarat-India

Ph no: 079 26608128, 95588-27732 Email: support@sunrobotics.in

Preface

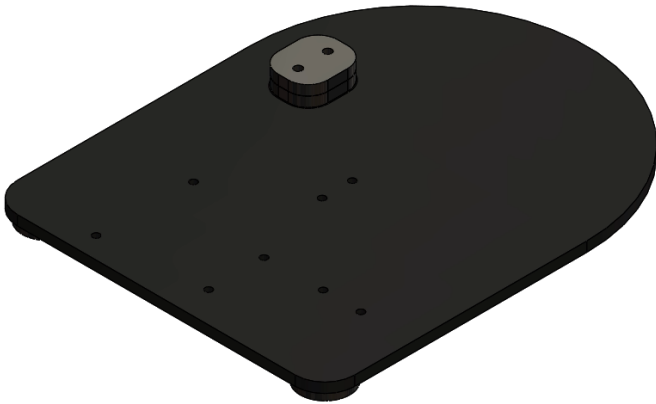
Sun Robotics is a technical service team of open source software and hardware. Dedicated to applying the Internet and the latest industrial technology in open source area, we strive to provide best hardware support and software service for general makers and electronic enthusiasts around the world. We aim to create infinite possibilities with sharing. No matter what field you are in, we can lead you into the electronic world and bring your ideas into reality.

This is a Professional Level kit for Arduino. Some common electronic components and sensors are included. Through the learning, you will get a better understanding of Arduino, and be able to make fascinating works based on Arduino.

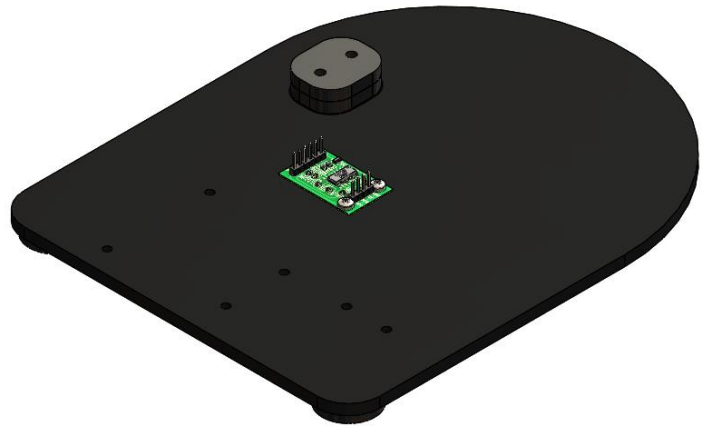
Contents

- Assemble All Modules with chassis
- Getting Started with Arduino
- Installing Arduino IDE and using the Uno R3 board
- About Arduino Uno R3 board
- Lesson- 1 LCD16x2 Keypad Shield Display using button with arduino
- Lesson- 2 Load cell Calibration Factor set for tare weight (Interface with HX711 Balance Module)
- Lesson- 3 Weight-20kg Display On LCD16x2 & serial terminal and also weight tare with RST button

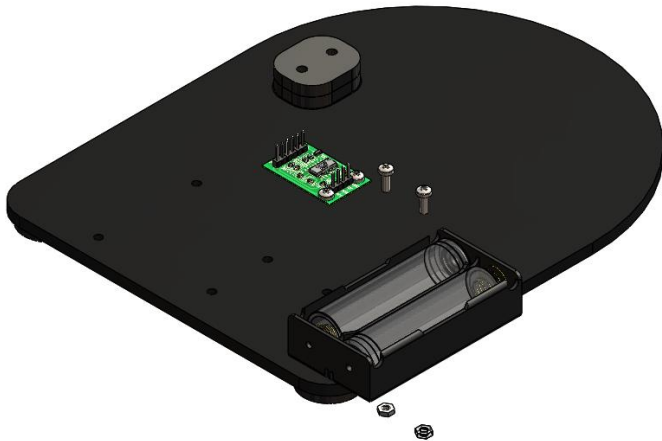
Assemble All Modules with Chassis



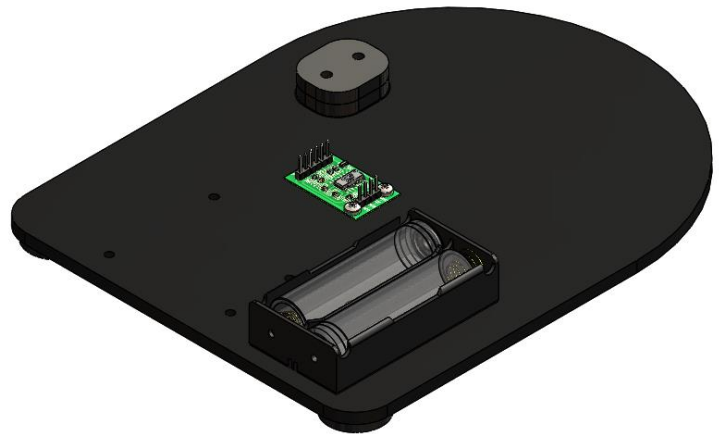
Step:-1



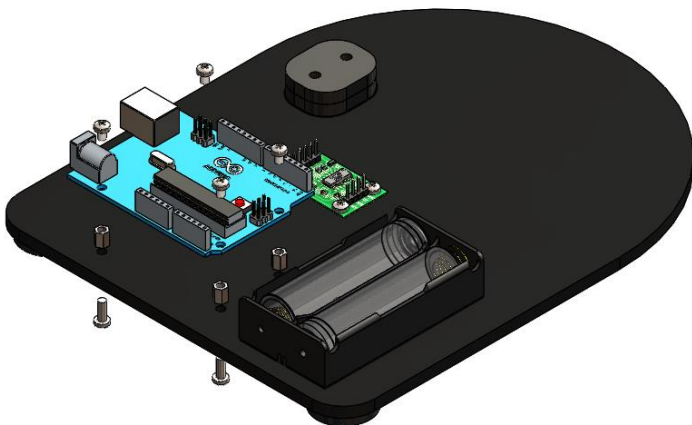
Step:-2



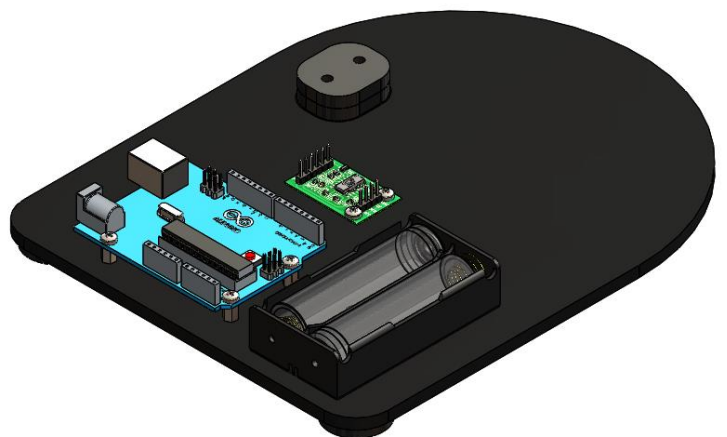
Step:-3



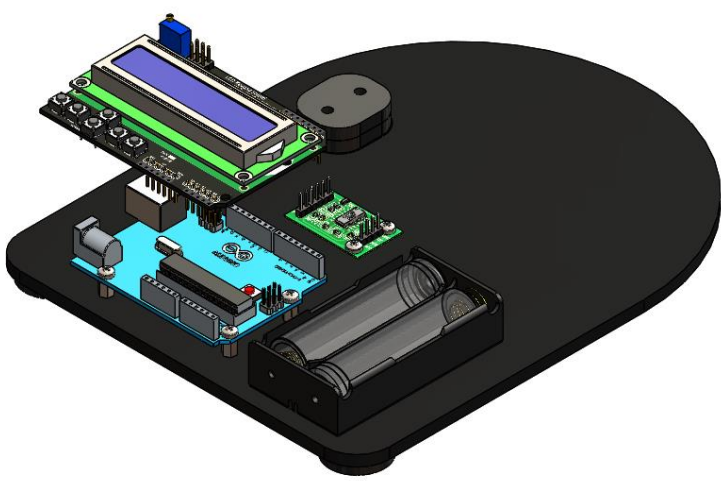
Step:-4



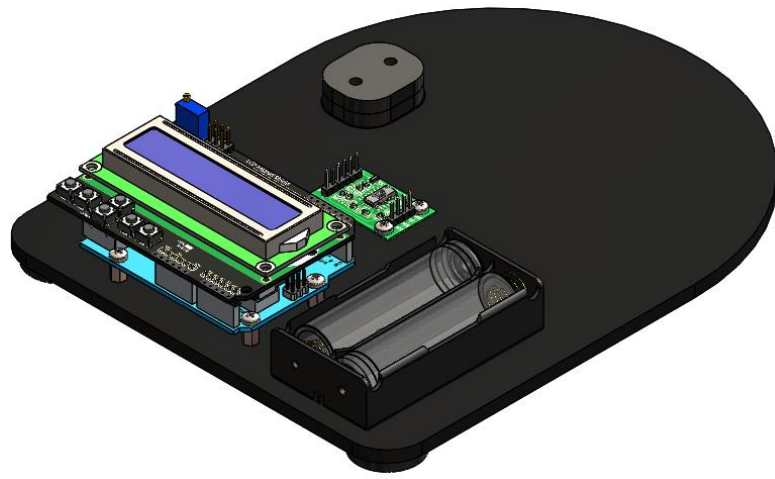
Step:-5



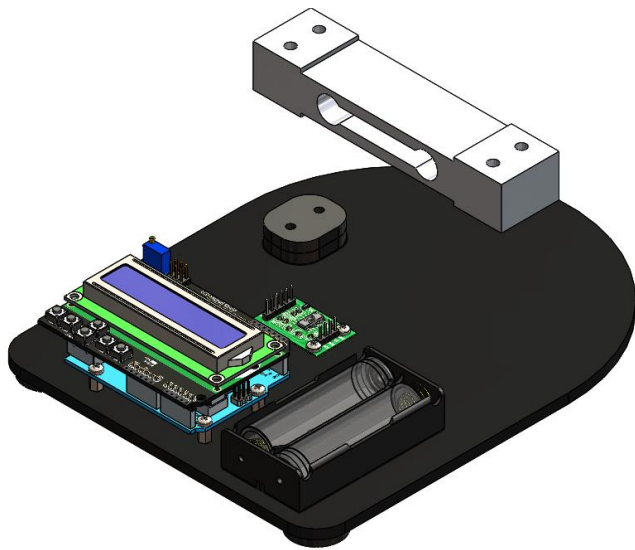
Step:-6



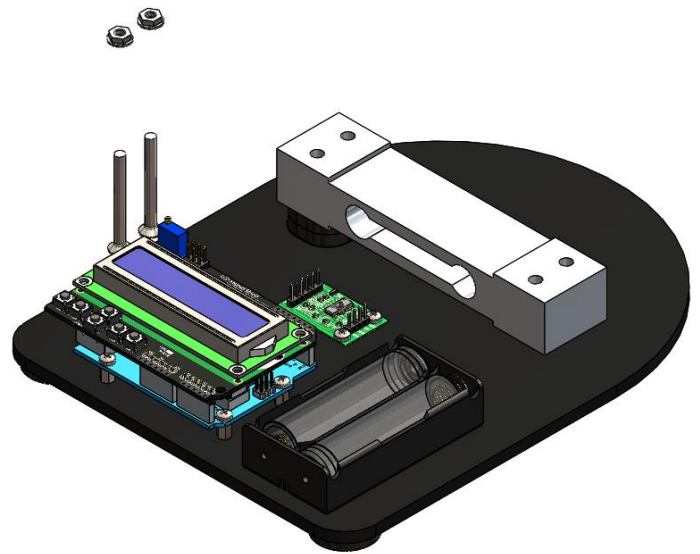
Step:-7



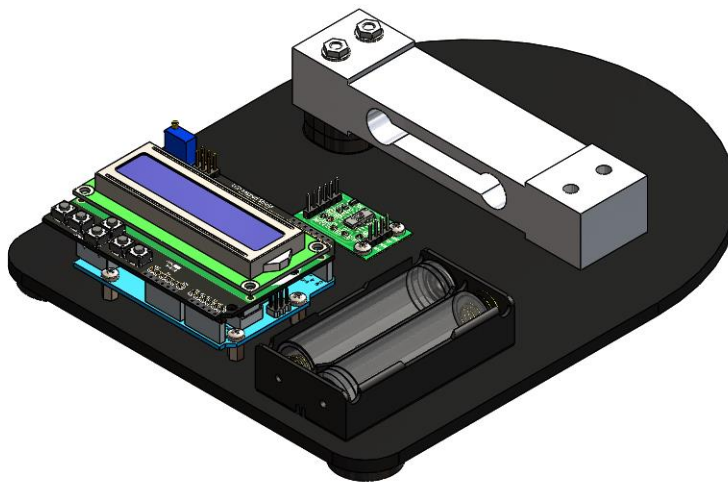
Step:-8



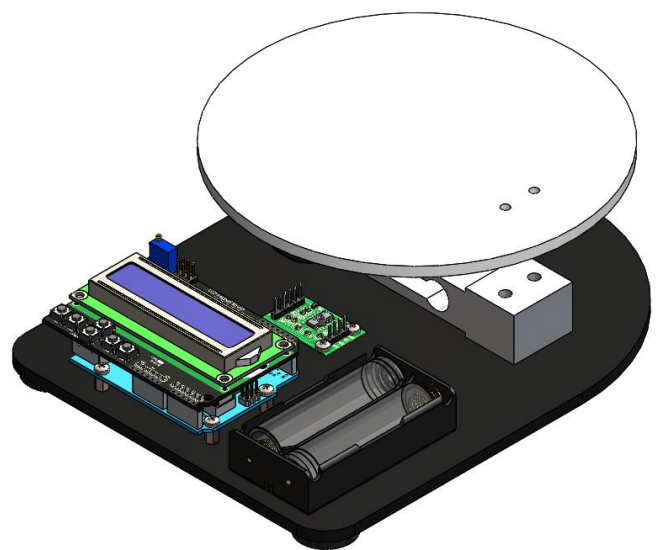
Step:-9



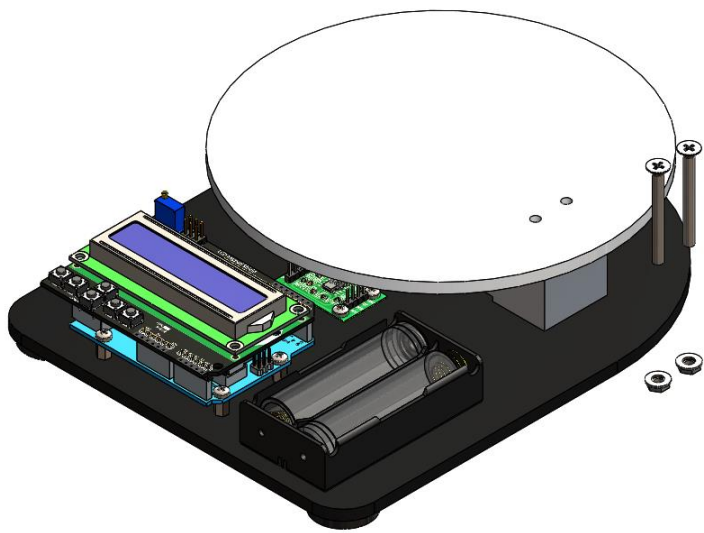
Step:-10



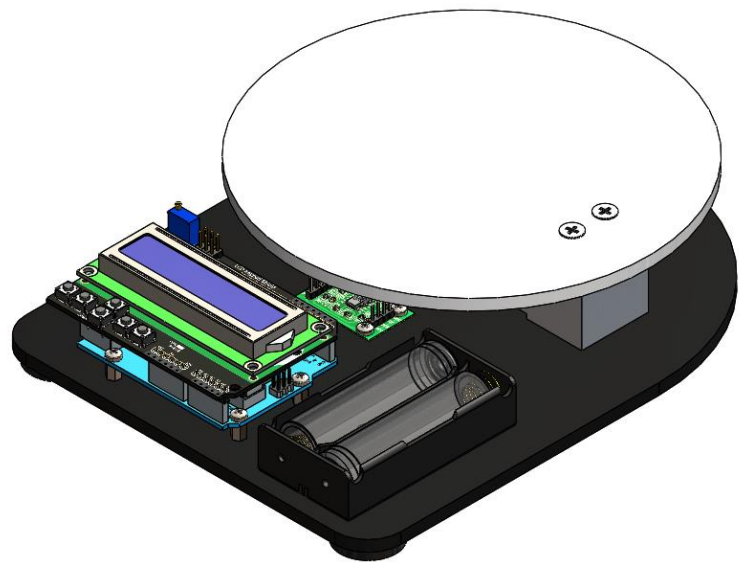
Step:-11



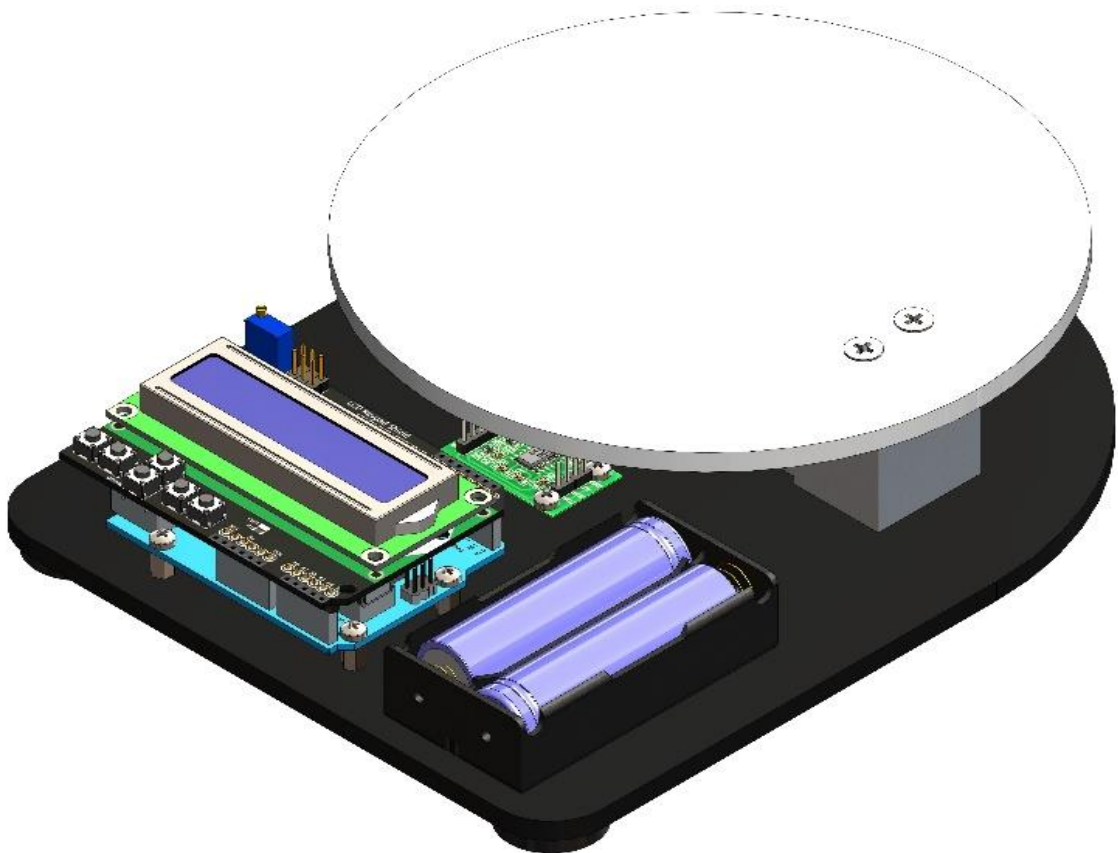
Step:-12



Step:-13



Step:-14



Step:-15

Getting Started with Arduino

What is an Arduino?

Arduino is an open-source physical computing platform designed to make experimenting with electronics more fun and intuitive. Arduino has its own unique, simplified programming language, a vast support network, and thousands of potential uses, making it the perfect platform for both beginner and advanced DIY enthusiasts.

www.arduino.cc

A Computer for the Physical World:

The friendly blue board in your hand (or on your desk) is the Arduino. In some ways you could think of Arduino as the child of traditional desktop and laptop computers. At its roots, the Arduino is essentially a small portable computer. It is capable of taking inputs (such as the push of a button or a reading from a light sensor) and interpreting that information to control various outputs (like a blinking LED light or an electric motor). That's where the term "physical computing" is born - an Arduino is capable of taking the world of electronics and relating it to the physical world in a real and tangible way. Trust us - this will all make more sense soon.

Arduino UNO SMD R3

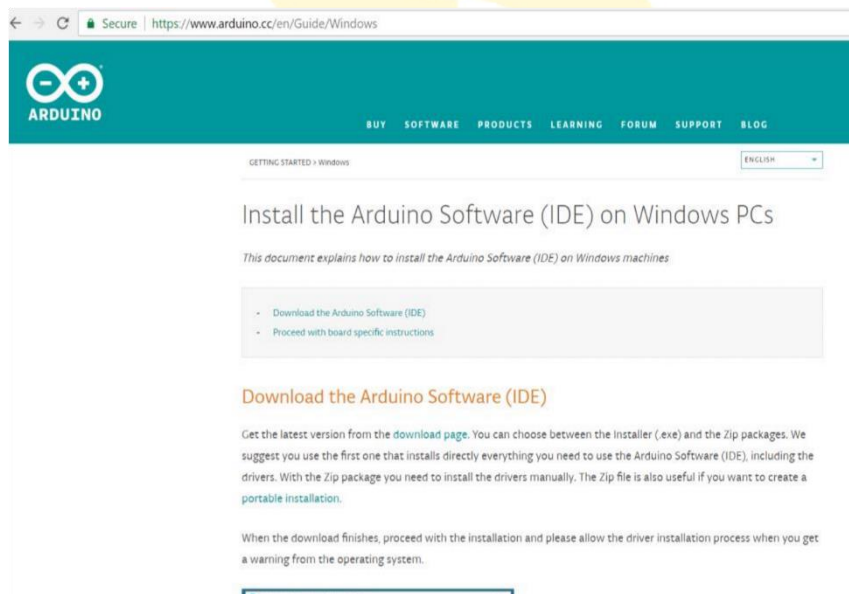
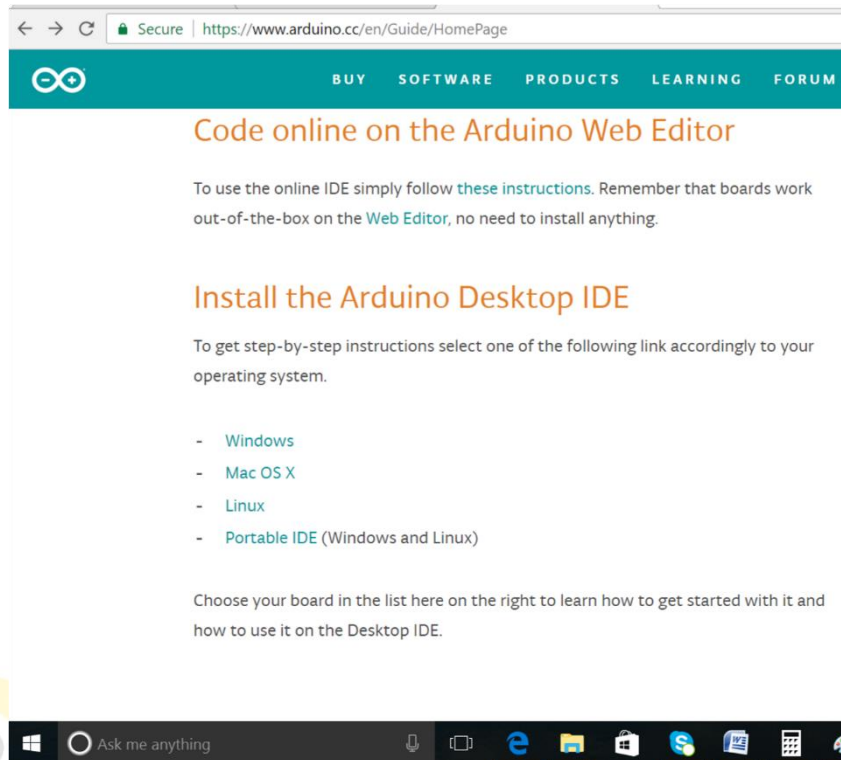
The Arduino Uno is one of several development boards based on the ATmega328. We like it mainly because of its extensive support network and its versatility. It has 14 digital input/output pins (6 of which can be PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. Don't worry, you'll learn about all these later.

Installing Arduino IDE and Using Uno R3 board

STEP-1: Download the Arduino IDE (Integrated Development Environment)

Access the Internet: In order to get your Arduino up and running, you'll need to download some software first from www.arduino.cc (it's free!). This software, known as the Arduino IDE, will allow you to program the Arduino to do exactly what you want. It's like a word processor for writing programs. With an internet-capable computer, open up your favorite browser and type in the following URL into the address bar:

www.arduino.cc/en/Main/Software

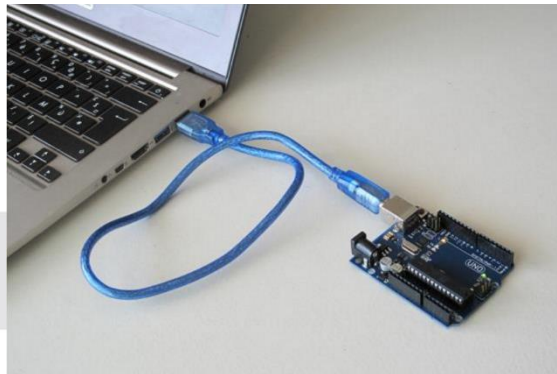


For different operating system platforms, the way of using Arduino IDE is different. Please refer to the following links: Windows User : <http://www.arduino.cc/en/Guide/Windows> Mac OS

User:<http://www.arduino.cc/en/Guide/MacOSX>Linux User<http://playground.arduino.cc/Learning/Linux> For more detailed information about Arduino IDE, please refer to the following link: <http://www.arduino.cc/en/Guide/HomePage>

STEP-2: Connect your Arduino Uno to your Computer:

Use the USB cable provided in the kit to connect the Arduino to one of your computer's USB inputs.



STEP-3: Install Drivers

Depending on your computer's operating system, you will need to follow specific instructions. Please go to the URLs below for specific instructions on how to install the drivers onto your Arduino Uno.

Windows Installation Process: Go to the web address below to access the instructions for installations on a Windows-based computer.

<http://arduino.cc/en/Guide/Windows>

Macintosh OS X Installation Process: Macs do not require you to install drivers. Enter the following URL if you have questions. Otherwise proceed to next page.

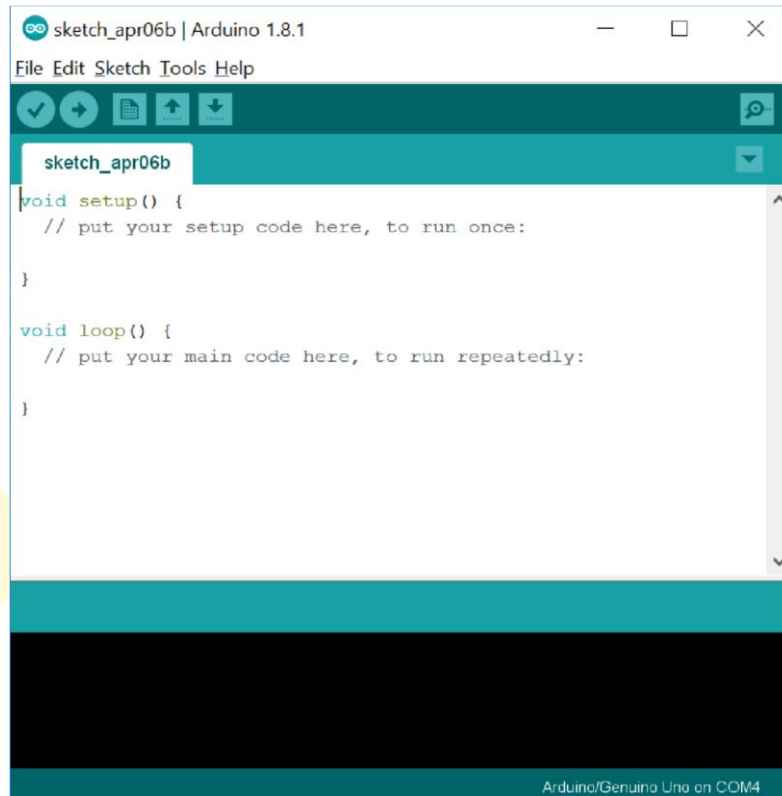
<http://arduino.cc/en/Guide/MacOSX>

Linux: 32 bit / 64 bit, Installation Process Go to the web address below to access the instructions for installations on a Linux-based computer.

<http://www.arduino.cc/playground/Learning/Linux>

STEP-4: Open the Arduino IDE

Open the Arduino IDE software on your computer. Poke around and get to know the interface. We aren't going to code right away, this is just an introduction. The step is to set your IDE to identify your Arduino Uno.



GUI (Graphical User Interface)



Verify

Checks your code for errors compiling it.



Upload

Compiles your code and uploads it to the configured board. See **uploading** below for Details .Note: If you are using an external programmer with your board, you can hold Down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer" New Creates a new sketch.



Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within The current window overwriting its content. Note: due to a bug in Java, this menu Doesn't scroll; if you need to open a sketch late in the list, use the File | Sketch book Menu instead.



Save

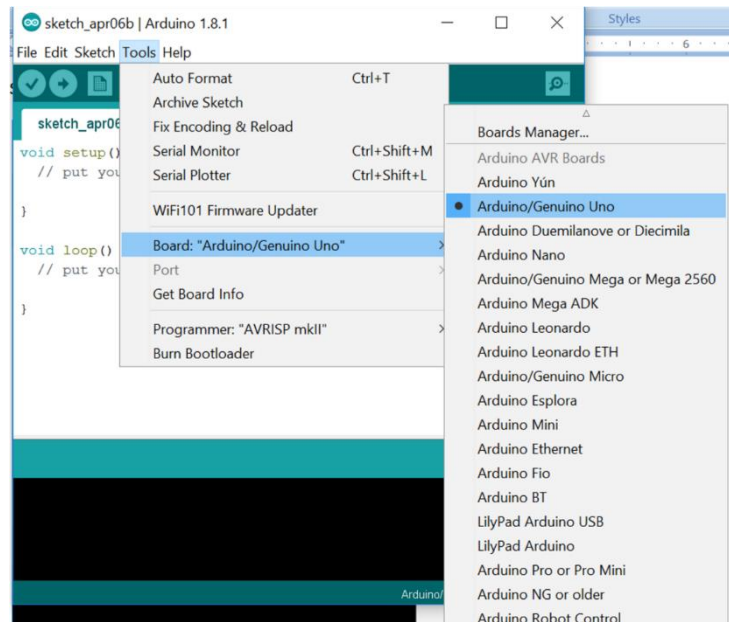
Saves your sketch.



Serial Monitor

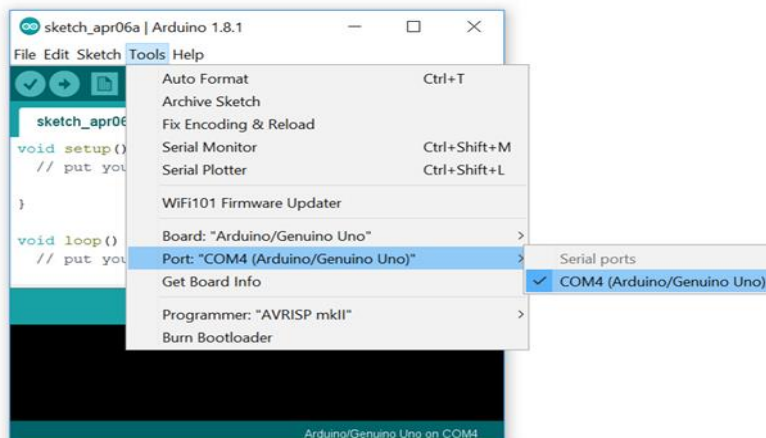
Opens the [serial monitor](#).

STEP-5: Select your board: Arduino Uno



STEP-6: Select your Serial Device

Windows: Select the serial device of the Arduino board from the Tools | Serial Port menu. This is likely to be com3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.



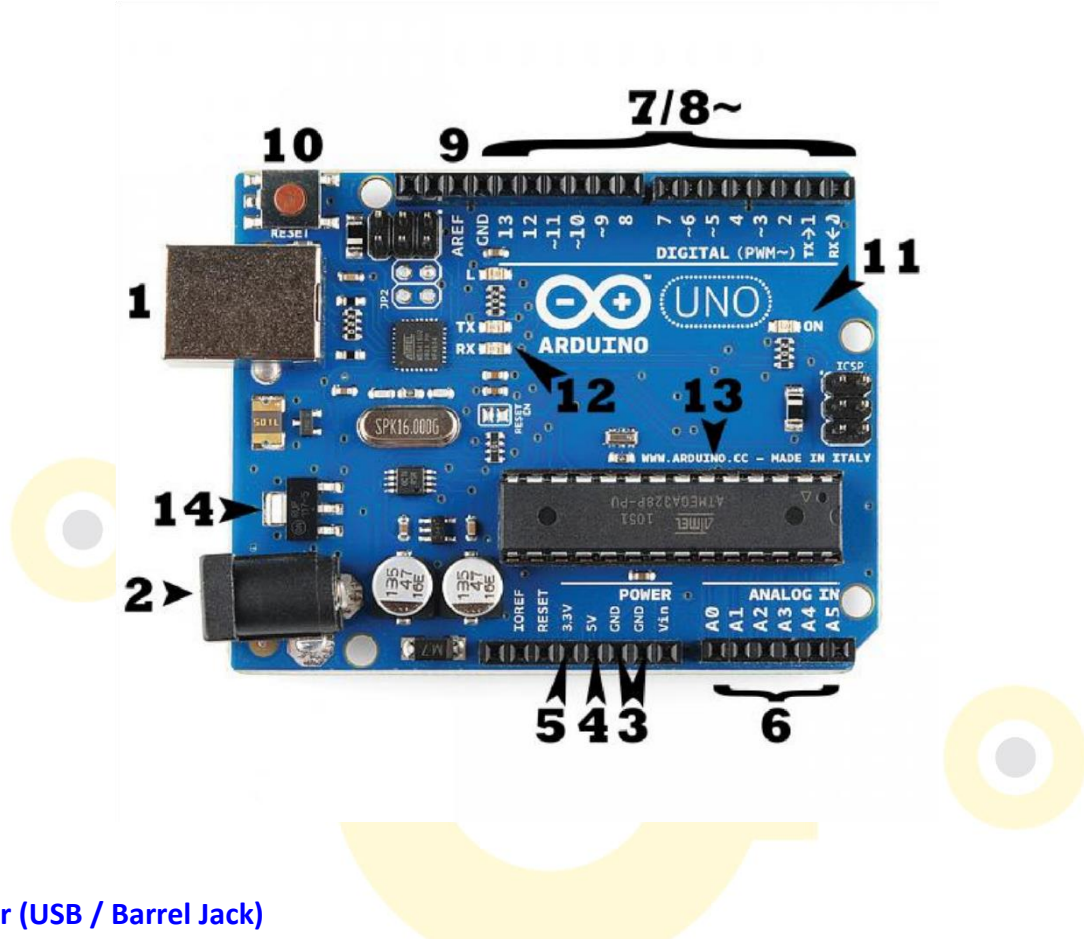
Mac OS: Select the serial device of the Arduino board from the Tools > Serial Port menu. On the Mac, this should be something with /dev/tty.usbmodem (for the Uno or Mega 2560) or /dev/tty.usbserial (for older boards) in it.

Linux: <http://playground.arduino.cc/Learning/Linux>

About Arduino Uno R3 board

What's on the board?

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduino have the majority of these components in common:



Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable coming from your computer or a wall power supply that is terminated in a barrel jack. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino. The recommended voltage for most Arduino models is between 6 and 12 Volts. Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire). They usually have black plastic 'headers' that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND (3):** Short for 'Ground'. There are several GND pins on the Arduino, any of which can be used to ground your circuit.
- **5V (4) & 3.3V (5):** As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6):** The area of pins under the 'Analog In' label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **Digital (7):** Across from the analog pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).
- **PWM (8):** You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO). These pins act as normal digital pins, but can also be used for something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now, think of these pins as being able to simulate analog output (like fading an LED in and out).
- **AREF (9):** Stands for Analog Reference. Most of the time you can leave this pin alone. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Just like the original Nintendo, the Arduino has a reset button (10). Pushing it will temporarily connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original Nintendo however, blowing on the Arduino doesn't usually fix any problems.

Power LED Indicator

Just beneath and to the right of the word "UNO" on your circuit board, there's a tiny LED next to the word 'ON' (11). This LED should light up whenever you plug your Arduino into a power source. If this light doesn't turn on, there's a good chance something is wrong. Time to re-check your circuit!

TX RX LEDs

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

Main IC

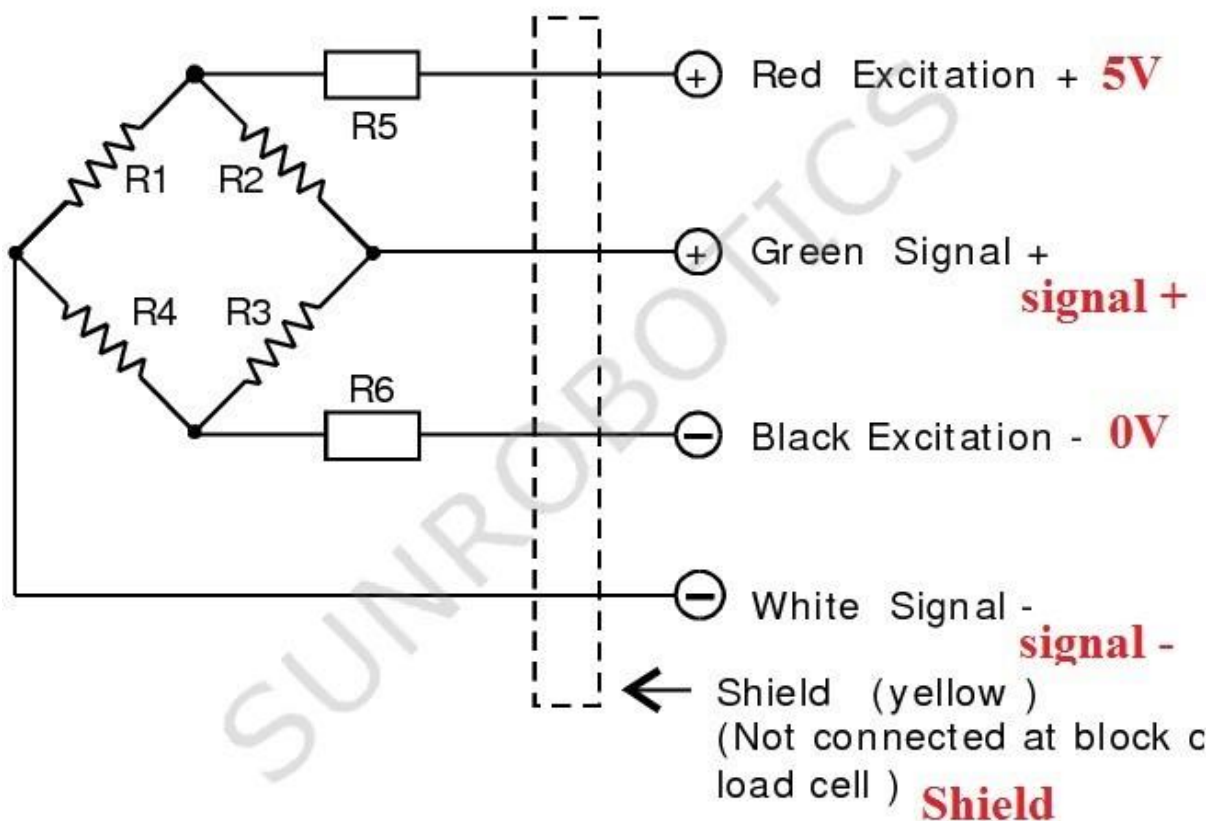
The black thing with all the metal legs is an IC, or Integrated Circuit (13). Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the AT mega line of IC's from the ATMEL company. This can be

important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

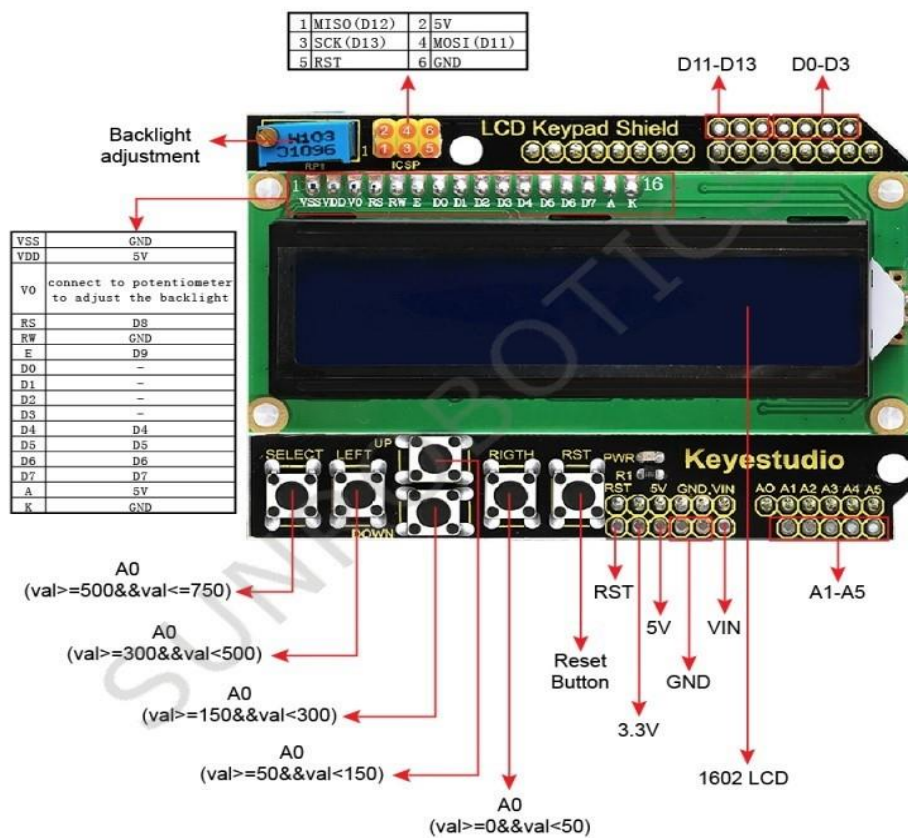
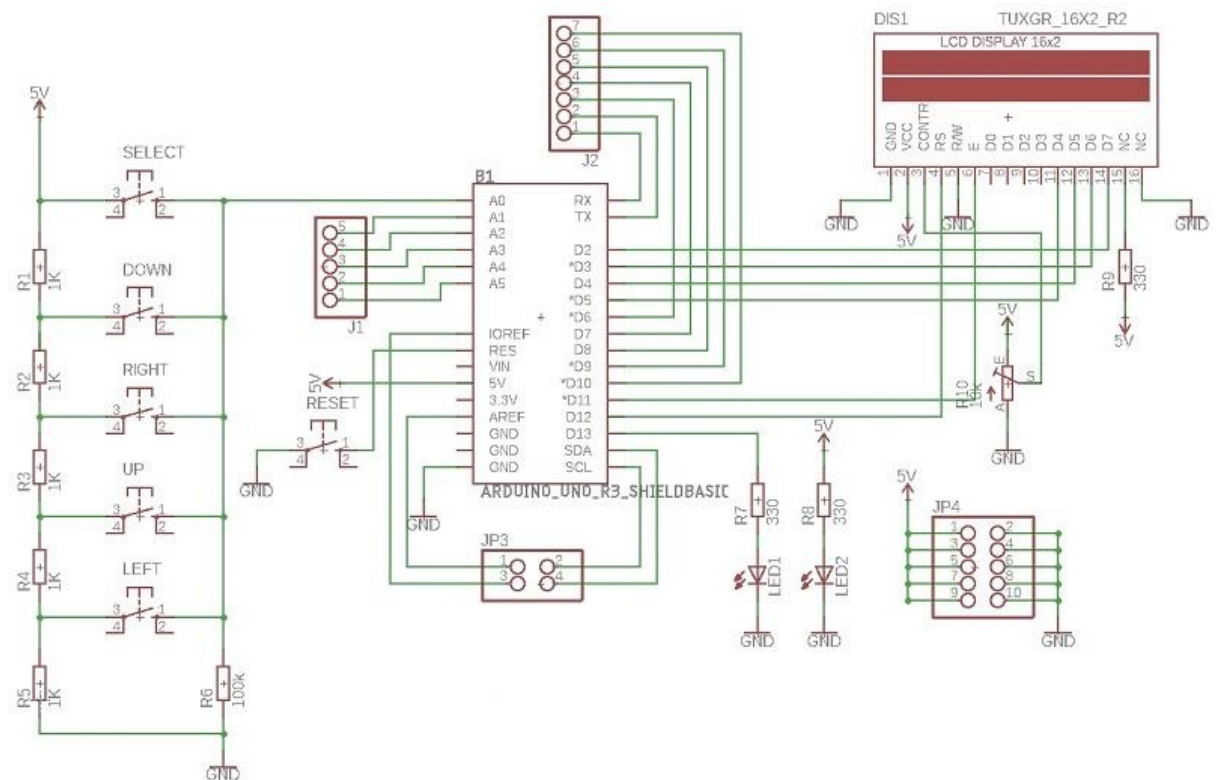
Voltage Regulator

The voltage regulator (14) is not actually something you can (or should) interact with on the Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

▪ Load Cell schematic:



■ LCD1602 Keypad Shield schematic:



Lesson 1 – LCD1602 Keypad Shield Display using button with arduino

Overview:

In this lesson, we will learn how to use a character display using LCD1602 Keypad Shield device on the Arduino platform. We first make the LCD1602 display a string "Left", "Right", "up", "down" on corresponding button pressed.

Components

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x LCD(16 x 2) Keypad Shield Display
- Jumper Wires

Principle

LCD1602 is a kind of character LCD display. The LCD has a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display. The interface consists of the following pins:

- A register select (RS) pin that controls where in the LCD's memory you're writing data to. You can select either the data register, which holds what goes on the screen, or an instruction register, which is where the LCD's controller looks for instructions on what to do next.
- A Read/Write (R/W) pin that selects reading mode or writing mode
- An Enable pin that enables writing to the registers
- 8 data pins (D0-D7). The state of these pins (high or low) is the bits that you're writing to a register when you write, or the values when you read.
- There are also a display contrast pin (Vo), power supply pins (+5V and Gnd) and LED Backlight (BKlt+ and BKlt-) pins that you can use to power the LCD, control the display contrast, and turn on or off the LED backlight respectively.

The process of controlling the display involves putting the data that form the image of what you want to display into the data registers, then putting instructions in the instruction register. The Liquid Crystal Library simplifies this for you so you don't need to know the low-level instructions. The Hitachi-compatible LCDs can be controlled in two modes: 4-bit or 8-bit. The 4-bit mode requires seven I/O pins from the Arduino, while the 8-bit mode requires 11 pins. For displaying text on the screen, you can do most everything in 4-bit mode.

Set LCD Contrast Using a pot which is interface on LCD board.

Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the “CODE” Folder

```
/*  
Arduino 2x16 LCD - Detect Buttons  
modified on 18 Feb 2019  
*/  
#include <LiquidCrystal.h>  
//LCD pin to Arduino  
const int pin_RS = 8;  
const int pin_EN = 9;  
const int pin_d4 = 4;  
const int pin_d5 = 5;  
const int pin_d6 = 6;  
const int pin_d7 = 7;  
const int pin_BL = 10;  
LiquidCrystal lcd( pin_RS, pin_EN, pin_d4, pin_d5, pin_d6, pin_d7);  
void setup() {  
  lcd.begin(16, 2);  
  lcd.setCursor(0,0);  
  lcd.print("--SunRobotics--");  
  lcd.setCursor(0,1);  
  lcd.print("Press Key:");  
}  
void loop() {
```

```
int x;  
x = analogRead (0);  
lcd.setCursor(10,1);  
if (x < 60) {  
    lcd.print ("Right ");  
}  
else if (x < 200) {  
    lcd.print ("Up  ");  
}  
else if (x < 400){  
    lcd.print ("Down ");  
}  
else if (x < 600){  
    lcd.print ("Left ");  
}  
else if (x < 800){  
    lcd.print ("Select");  
}  
}
```

Step 3: Compile the program and upload to Arduino UNO board

Step 4: Output (Working)

- When Up key pressed LCD display in second line "Press Key: UP"
- When Down key pressed LCD display in second line "Press Key: Down"
- When Left key pressed LCD display in second line "Press Key: Left"
- When Right key pressed LCD display in second line "Press Key: Right"

Lesson 2 – Load cell Calibration Factor set for tare weight (Interface with HX711 Balance Module)

Overview:

In tutorial of an Arduino project, which lets you turn your Arduino to a “Weight Scale”, with a 10 kg load cell and a HX711 load cell amplifier breakout circuit, here, first calibration factor set for known weight value.

Components:

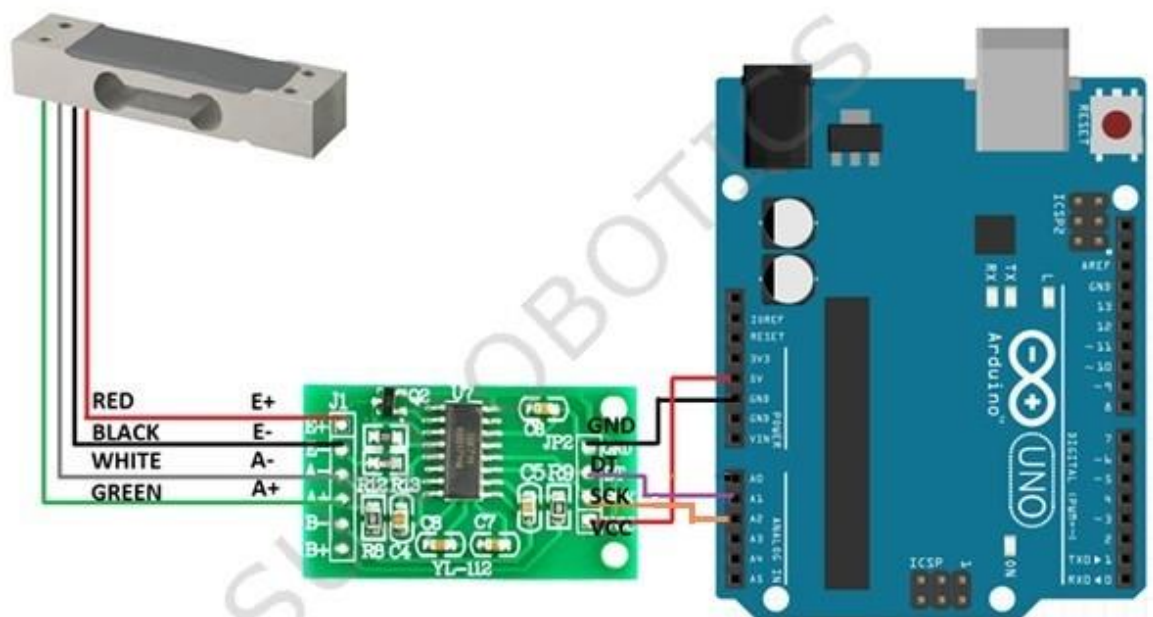
- 1 x Arduino UNO
- 1 x USB Cable
- 1 x HX711 Balance Sensor
- 1 x Load cell with Platform
- Jumper Wires

Principle

A load cell is a transducer that converts a physical force to an electric signal. There are various types of load cells, hydraulic load cells, pneumatic load cells, magnetostrictive load cells, Piezoresistive load cells and strain gauge load cells, which we'll use for our project. Strain gauge load cells are usually set up in Z shape forms so that the applied torque can be measured by the bending of the gauges. Applied weight (or torque you can say) changes the electrical resistance of the gauges proportional to the force. Load cells are used everywhere where scaling is needed: mechanical balancing applications, processing plants, industrial and platform scales, tank dispensing systems precision laboratory balances.

Procedure:

Step 1: Build the circuit




```

#include "HX711.h"

#define DOUT A1
#define CLK A2
HX711 scale;

//float calibration_factor = -146940; //for 3 kg weighing scale.
//float calibration_factor = 192100; //for 10kg weighing scale.
float calibration_factor = -49051; //It worked for my 40Kg max scale setup
float weight;

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press a,s,d,f to increase calibration factor by 10,100,1000,10000 respectively");
  Serial.println("Press z,x,c,v to decrease calibration factor by 10,100,1000,10000 respectively");
  scale.set_scale();
  scale.tare(); //Reset the scale to 0
  long zero_factor = scale.read_average(); //Get a baseline reading
  Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in
  permanent scale projects.
  Serial.println(zero_factor);
}

void loop() {
  scale.set_scale(calibration_factor); //Adjust to this calibration factor

  Serial.print("Reading: ");
  //weight = scale.get_units();
  weight = (0.454 * scale.get_units());
  Serial.print(weight, 3);
  Serial.print(" Kg"); //Change this to kg and re-adjust the calibration factor if you follow SI units
  like a sane person
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  if (Serial.available())
  {
    char temp = Serial.read();
    if (temp == '+' || temp == 'a')

```

```

    calibration_factor += 10;
else if (temp == '-' || temp == 'z')
    calibration_factor -= 10;
else if (temp == 's')
    calibration_factor += 100;
else if (temp == 'x')
    calibration_factor -= 100;
else if (temp == 'd')
    calibration_factor += 1000;
else if (temp == 'c')
    calibration_factor -= 1000;
else if (temp == 'f')
    calibration_factor += 10000;
else if (temp == 'v')
    calibration_factor -= 10000;
else if (temp == 't')
    scale.tare(); //Reset the scale to zero
}
}

```

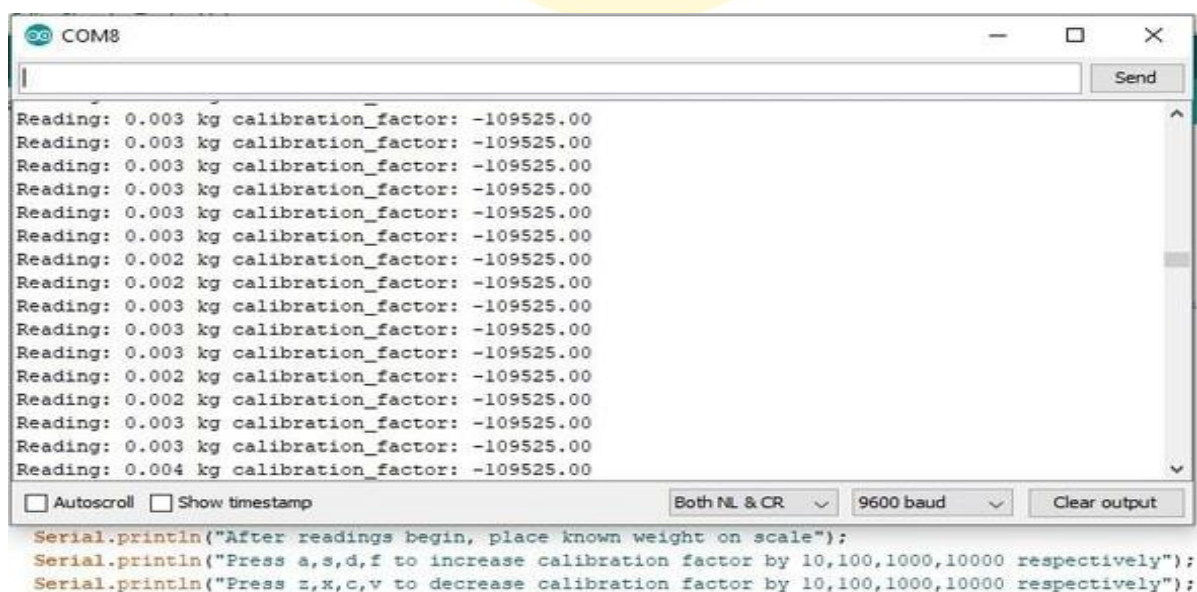
Step 2: Program: Open /Copy the code from the “CODE” Folder

Step 3: Compile the program and upload to Arduino UNO board

Step 4: Output in serial monitor in kilogram

Once you upload the **calibration code**, open the serial monitor and adjust your scale factor with known weight until you see the correct readings. Press a,s,d,f to increase calibration factor by 10,100,1000,10000 respectively. Press z,x,c,v to decrease calibration factor by 10,100,1000,10000 respectively.

Once you see the placed weight is the same as shown weight note down the **calibration factor** and use it in the final code for Weighing Scale.



The screenshot shows the Arduino Serial Monitor window titled "COM8". The output displays a series of readings: "Reading: 0.003 kg calibration_factor: -109525.00" repeated 15 times. The final reading is "Reading: 0.004 kg calibration_factor: -109525.00". At the bottom, there are instructions: "After readings begin, place known weight on scale"; "Press a,s,d,f to increase calibration factor by 10,100,1000,10000 respectively"; and "Press z,x,c,v to decrease calibration factor by 10,100,1000,10000 respectively". The serial monitor settings are set to "Both NL & CR", "9600 baud", and "Clear output" is available.

Lesson- 3 Weight-20kg Display On LCD16x2 & serial terminal and also weight tare with RST button

Overview:

In tutorial of an Arduino project, which lets you turn your Arduino to a “Weight Scale”, with a 20 kg load cell and a HX711 load cell amplifier breakout circuit, that measures the weight of an object. Weight display on LCD1602 Keypad Shield device in Kilogram. Weight also tare using “RST” button which on LCD16X2 Keypad Shield device.

Components:

- 1 x Arduino UNO
- 1 x USB Cable
- 1 x HX711 Balance Sensor
- 1 x Load cell with Platform
- 1 x LCD(16 x 2) Keypad Shield Display

Working:

Object's Weight Display on LCD16X2 in kg after weight Calibration.

Procedure:

Step 1: Build the circuit



Step 2: Program: Open /Copy the code from the "CODE" Folder

```
#include "HX711.h"
#include <LiquidCrystal.h>

//set calibration factor known weight measurement if you not get perfect weight

#define calibration_factor -90230 // for 20kg load cell

#define LOADCELL_DOUT_PIN A1
#define LOADCELL_SCK_PIN A2

//LCD pin to Arduino
const int pin_RS = 8;
const int pin_EN = 9;
const int pin_d4 = 4;
const int pin_d5 = 5;
const int pin_d6 = 6;
const int pin_d7 = 7;
const int pin_BL = 10;

float weight;
float real_weight;
float minimum_weight = 0.10; // minimum weight 0.10 kg
float average = 0;
float total_avr = 0;
bool data_tx_flag = false;

LiquidCrystal lcd( pin_RS, pin_EN, pin_d4, pin_d5, pin_d6, pin_d7);
HX711 scale;

void setup() {
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.setCursor(0, 0);
  lcd.print("- Object Weight-");
  lcd.setCursor(0, 1);
  lcd.print("in Kg.");
  // load cell code below
  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale(calibration_factor); //This value is obtained by using the
  SparkFun_HX711_Calibration sketch
  scale.tare(); //Assuming there is no weight on the scale at start up, reset the scale to 0
}
float weightReadInstance()
{
  //load cell data weight in kg
```

```

weight = (0.454 * scale.get_units());
if (weight > minimum_weight) {
    real_weight = weight;
}
else {
    real_weight = 0.00;
}
return real_weight;
}
void loop() {
    average = weightReadInstance();
    if (average > 0.01) {
        real_weight = (0.454 * scale.get_units());
        if ((real_weight > total_avr + 0.020) || (data_tx_flag == true)) {
            for (int i = 0; i < 10; i++) {
                total_avr = (0.454 * scale.get_units());
                delay(10);
            }
            Serial.println(total_avr, 3);
            data_tx_flag = false;
        }
        else if (real_weight < total_avr - 0.020) {
            data_tx_flag = true;
        }
    }
    // if ( analogRead(A0) == 721) { // SELECT Button presss send data serially
    //   Serial.println(total_avr, 3);
    //   delay(2000); // 2-second wait after release
    // }
    else if (average < 0.01) {
        total_avr = 0.00;
    }
    lcd.setCursor(7, 1);
    lcd.print (total_avr, 3);
    lcd.print (" Kg ");
    delay(100);
}

```

Step 3: Compile the program and upload to Arduino UNO board

Step 4: Output in LCD Display in kilogram

THANK YOU!



SunRobotics
Parts for Every Idea!

www.sunrobotics.in