

EP1000

Sensors

Input Devices - Sensors

- Sensors are input devices
 - Measure some physical quantity (touch, light, heat etc)
 - Changes are slow (compared to computational power)
 - Most readings are analog by nature
 - Requires conversion to digital for processing
- Methods of reading sensors
 - Data is always available (e.g. temperature)
 - Polling to check whether data is available from sensor
 - Triggering – sensor will send a signal indicating data

Typical Sensors

Physical Quatity	Sensor	Typical devices
Heat	Thermal probe	LM35 , DHT11 , DS18B20
Light	Light-sensitive transistor	LDR-5516 , Light detectors
Sound	Microphones	Sound Sensor KY-038
Distance	Ultrasonic distance measurer	HC-SR04
Touch	Capacitive touch plate	Touch switches
Movement	Infra-red movement detector	HC-SR501 PIR , RCWL-0516
Water (Humidity)	Humidity sensor	DHT-11 , Water level sensor
Time	Real-time Clocks	DS3231 , DS1302
Weight	Load Cell	LWC with HX711 ADC
Video	Video Camera	OV7670 , Pixy2

Ref:

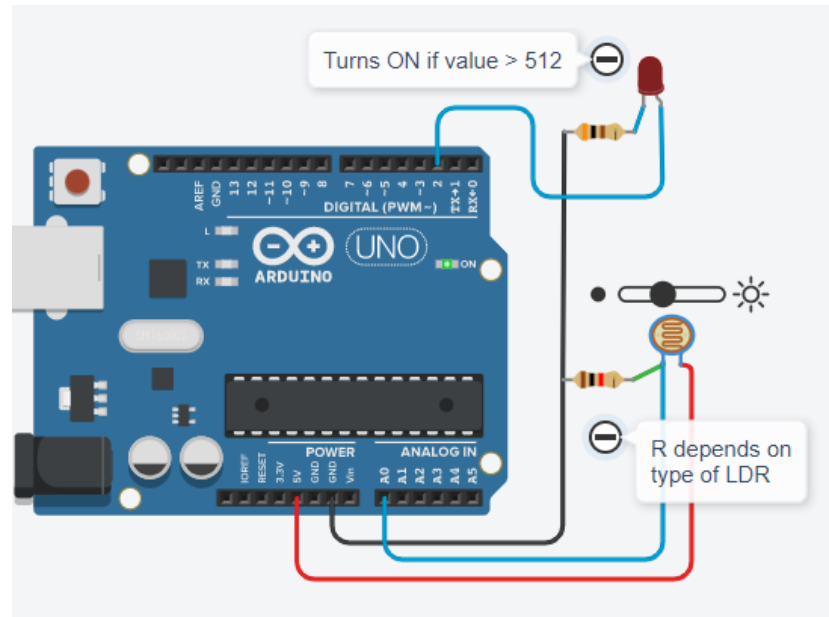
- [Arduino - 245 Sensor Projects](#)
- [Instructables – 37 in 1 Sensor Kit Explained](#)
- [Bas On Tech – Arduino Tutorials](#)

Measure Light Intensity

```

1  const int LED = 2;
2
3
4  void setup()
5  {
6      Serial.begin(115200);
7      pinMode(LED, OUTPUT);
8  }
9
10 void loop()
11 {
12     int value = analogRead(A0);
13     Serial.println(value);
14     if (value > 512)
15         digitalWrite(LED, HIGH);
16     else
17         digitalWrite(LED, LOW);
18 }

```



[Light-dependent Resistor \(LDR 5516\)](#)
Read the equivalent analog voltage

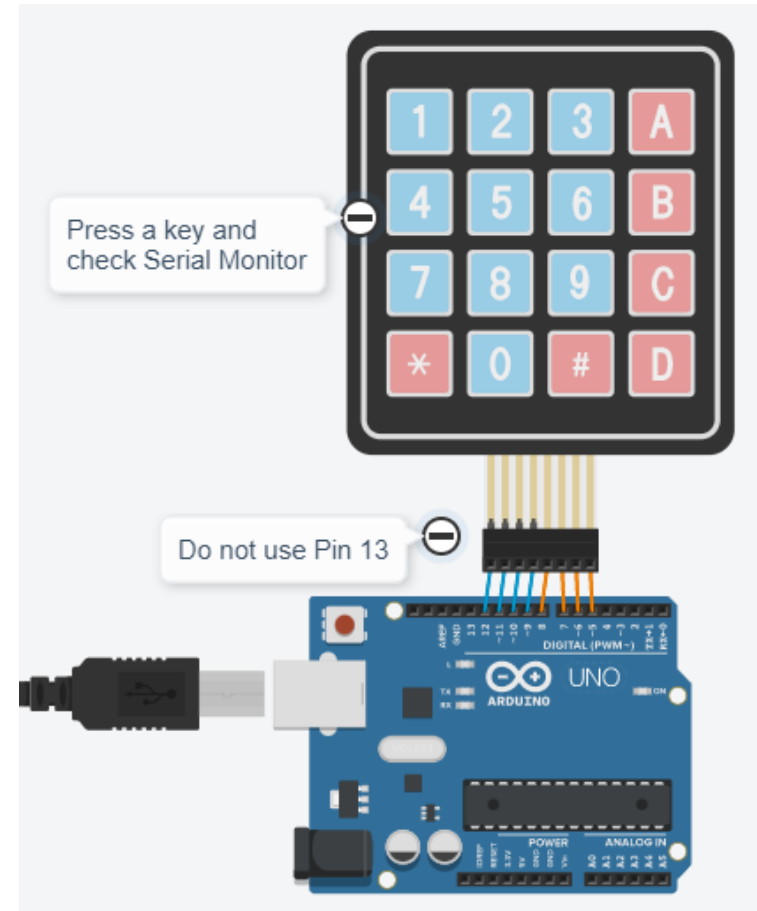
Matrix Keypad with Library

[KeyPad Library](#)

```

3 // Use the Keypad Library
4 #include <Keypad.h>
5 // Define the size
6 const byte ROWS = 4; //four rows
7 const byte COLS = 4; //four columns
8 //define the symbols on the buttons of the keypads
9 char hexaKeys[ROWS][COLS] = {
10   {'1','2','3','A'},
11   {'4','5','6','B'},
12   {'7','8','9','C'},
13   {'*','0','#','D'} };
14 // uno pin connections
15 byte rowPins[ROWS] = {12, 11, 10, 9};
16 byte colPins[COLS] = { 8, 7, 6, 5};
17 //initialize an instance of class NewKeypad
18 Keypad myKeypad = Keypad( makeKeymap(hexaKeys),
19   rowPins, colPins,ROWS, COLS);
20 void setup(){
21   Serial.begin(9600);
22   Serial.println("Starting...");
23 }
24
25 void loop(){
26   // read the keypad
27   char key = myKeypad.getKey();
28   // if valid key, output it
29   if (key){
30     Serial.println(key);
31   }
32 }

```

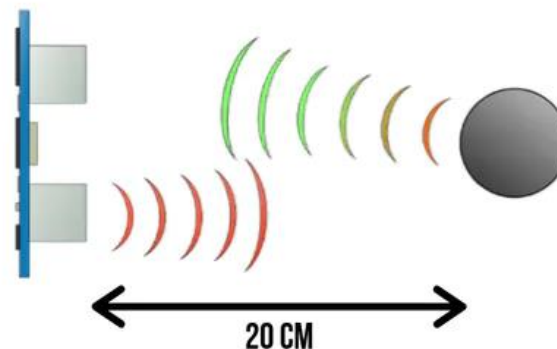


Simulation only works with some libraries

Ref: [Libraries with TinkerCAD](#)

Measuring Distance

- [Ultrasonic Sensor HC-SR04](#)
- Measures distances between 2cm to 400cm without contact using sound (ultrasonic)
- Requires a trigger and an input (2 digital pins)
- Lots of tutorials:
 - [Arduino.cc](#)
 - [Instructables](#)



SPEED OF SOUND:
 $v = 340 \text{ m/s}$
 $v = 0.034 \text{ m/s}$

TIME = DISTANCE/SPEED
 $t = s/v = 20/0.034$
 $= 588 \text{ us}$
 $s = t \times 0.034/2$

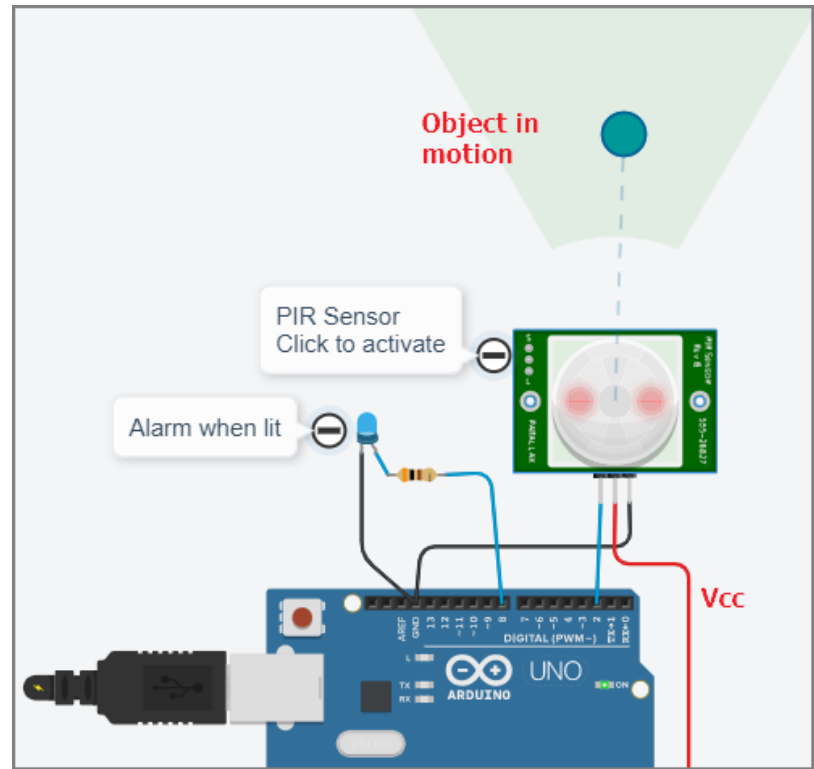
Measuring Distance – SR04

```
24
25 void loop() {
26   // Clears the trigPin condition
27   digitalWrite(trigPin, LOW);
28   delayMicroseconds(2);
29   // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
30   digitalWrite(trigPin, HIGH);
31   delayMicroseconds(10);
32   digitalWrite(trigPin, LOW);
33   // Reads the echoPin, returns the sound wave travel time in microseconds
34   duration = pulseIn(echoPin, HIGH);
35   // Calculating the distance
36   distance = duration * 0.034 / 2; // Speed of sound wave divided by 2
37   // Displays the distance on the Serial Monitor
38   Serial.print("Distance: ");
39   Serial.print(distance);
40   Serial.println(" cm");
41 }
```

- LOW pulse (10 mS) is used to trigger the sensor
- Return pulse is measured using pulseIn(), distance is proportional to pulse length

PIR Motion Sensor HC-SR501

- [Passive Infra Red](#)
- Detects motion
 - Adjust Sensitivity
 - Wait at least 15 s
- No Library required, 1 digital I/O input pin for status.
- Check pin
 - LOW no motion
 - HIGH motion detected
- Cannot measure distance

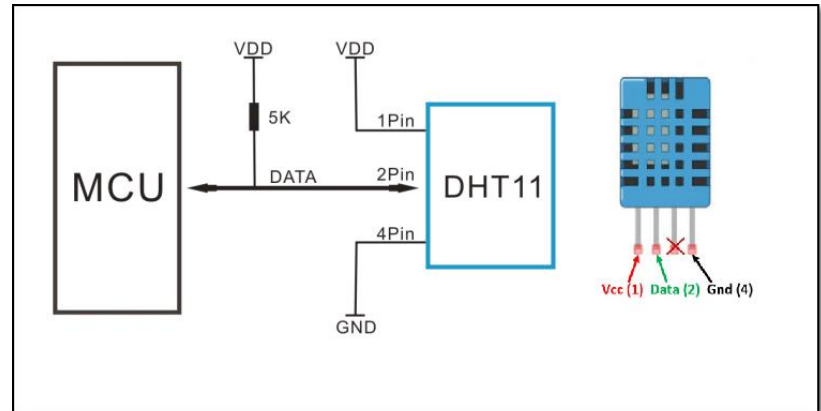


[Motion detection using PIR HC-SR501](#)

Better alternative: [RCWL-0516 Microwave Proximity Sensor](#)

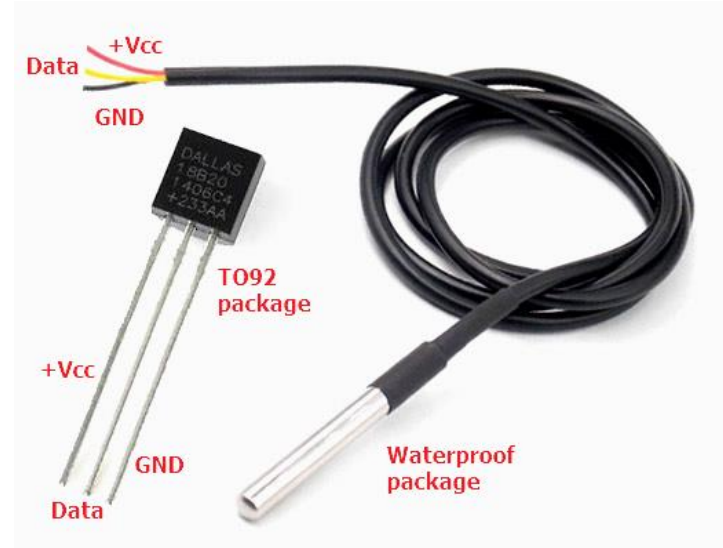
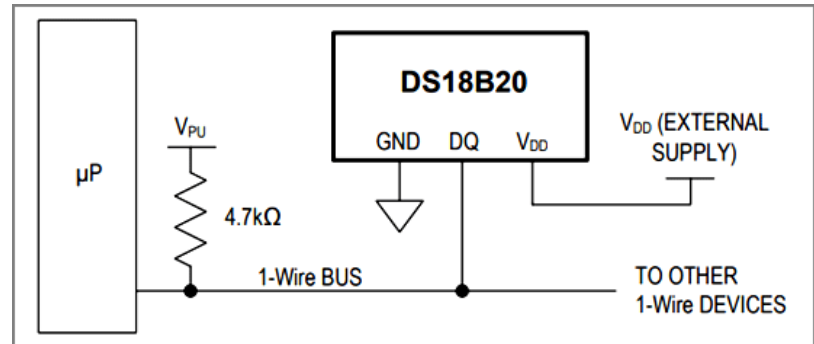
Room Temperature & Humidity

- [DHT-11](#) Temperature and Humidity sensor
- 20~80% humidity, 0~50°C
1 Hz sampling rate.
- [Library from Adafruit](#) (install both):
 - TinyDHT
 - TinyWire
- Requires 1 digital I/O pin
- Better results, accuracy with the DHT-22, however, 2~3x more expensive



Higher Temperatures: DS18B20

- Specifications:
 - Temperature range: $-55^{\circ}\text{C} \sim 125^{\circ}\text{C}$, Accuracy: $\pm 0.5^{\circ}\text{C}$
 - Communication: 1Wire
 - Sampling: 750mS at 12bit
- Library: [DS18B20 RT Arduino Temperature Control Library](#)
 - Minimal functions, simple
 - 1 sensor per MCU pin
- Uses: temperature sensing in hard environments, liquids away from processing unit

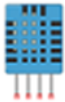







Reading the DS18B20

- Libraries:
 - [DS18B20_RT](#) sensor library
 - [OneWireNG](#) communications library
- Library only provides minimal functions
 - Instantiate object
 - Trigger sensor read
 - check data ready
 - Read temperature °C

```
3 // Libraries
4 #include <OneWire.h>
5 #include <DS18B20.h>
6
7 #define ONE_WIRE_BUS 2 // data pin
8
9 // create objects
10 OneWire oneWire(ONE_WIRE_BUS);
11 DS18B20 sensor(&oneWire);
12
13 void setup(void)
14 {
15     Serial.begin(115200);
16     sensor.begin(); // initialise
17 }
18
19
20 void loop(void)
21 {
22     // read
23     sensor.requestTemperatures();
24     // wait until sensor is ready
25     while (!sensor.isConversionComplete());
26     // results
27     Serial.print("Temp: ");
28     Serial.println(sensor.getTempC());
29 }
30
```

Temperature Sensor Comparison

Sensor	DHT11	DHT22 (AM2302)	LM35	DS18B20	BME280	BMP180
						
Measures	Temperature Humidity	Temperature Humidity	Temperature	Temperature	Temperature Humidity Pressure	Temperature Pressure
Communication protocol	One-wire	One-wire	Analog	One-wire	I2C SPI	I2C
Supply voltage	3 to 5.5V DC	3 to 6V DC	4 to 30 V DC	3 to 5.5V DC	1.7 to 3.6V (for the chip) 3.3 to 5V for the board	1.8 to 3.6V (for the chip) 3.3 to 5V for the board
Temperature range	0 to 50°C	-40 to 80°C	-55 to 150°C	-55 to 125°C	-40 to 85°C	0 to 65°C
Accuracy	+/- 2°C (at 0 to 50°C)	+/- 0.5°C (at -40 to 80°C)	+/-0.5°C (at 25°C)	+/-0.5°C (at -10 to 85°C)	+/-0.5°C (at 25°C)	+/-0.5°C (at 25°C)
Support (Arduino IDE)	Adafruit DHT Library Adafruit Unified Sensor Library	Adafruit DHT Library Adafruit Unified Sensor Library	analogRead()	DallasTemperature OneWire	Adafruit BME280 library Adafruit Unified Sensor Library	Adafruit BME085 Adafruit Unified Sensor Library

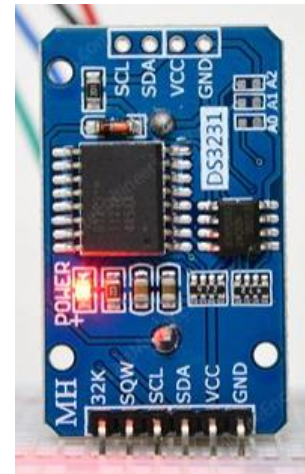
Real Time Clock Modules

- Uno has no RTC to keep track of time.
- Use RTCLib
- Use independent [RTC Modules](#)
 - [DS1307](#)
 - Has a 32KHz crystal oscillator (slightly off)
 - Has provision for DS18B20 connection
 - Has battery backup
 - [DS3231](#)
 - Uses I2C communications
 - Uses temperature controlled oscillator
 - Has battery backup

Real Time Clock Modules

Uno does not have a RTC module to keep accurate time (powerd off)
Use a RTC Clock Module and [Adafruit RTCLib](#)

- [DS1307](#)
 - Uses OneWire communications
 - Has a 32KHz crystal oscillator
 - Has provision for DS18B20 connection
 - Has battery backup
- [DS3231](#)
 - Uses I2C communications
 - Uses temperature controlled oscillator
 - Has battery backup



References:

- [Dronebot workshop](#)
- [Last Minute Engineers](#)
- [Arduino Project Hub](#)
- [Instructables](#)

EP1000

Sensors

End