EP1000

Sensors

Input Devices – Sensors

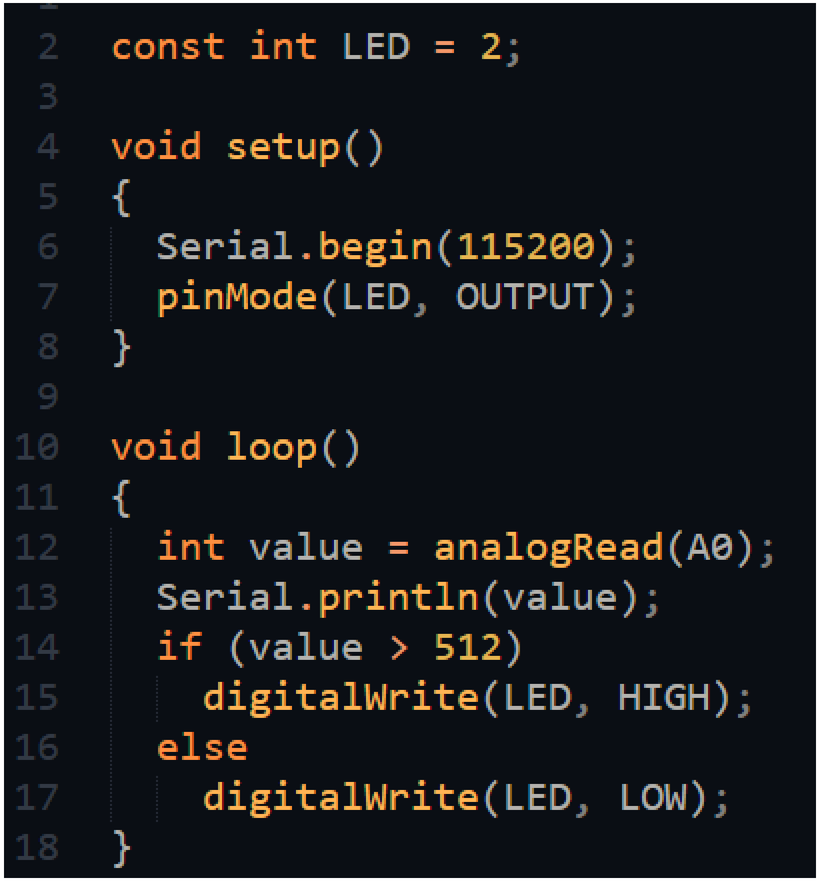
- Sensors are input devices
  - Measure some physical quantity (touch, light, heat etc)
  - Changes are slow (compared to computational power)
  - Most readings are analog by nature
  - Requires conversion to digital for processing
- Methods of reading sensors
  - Data is always available (e.g. temperature)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

Typical Sensors

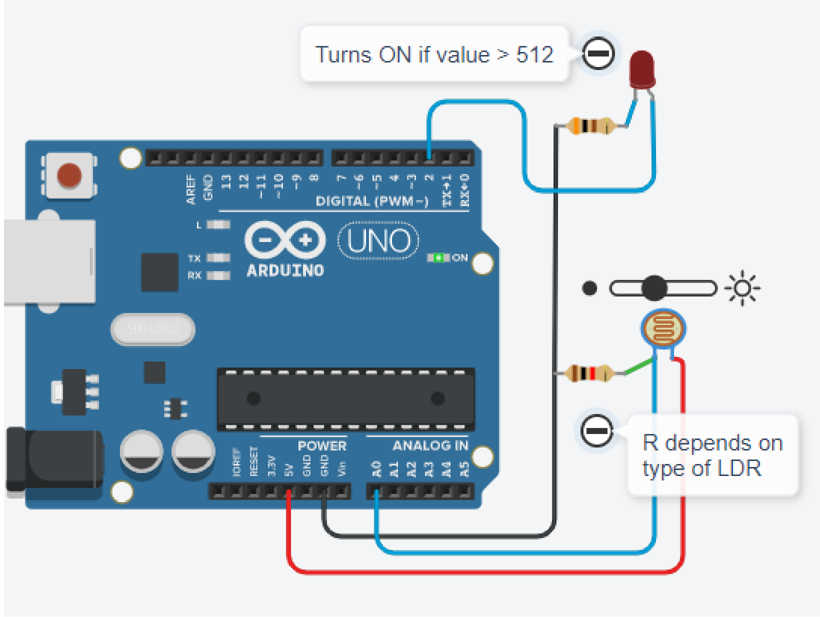
Physical Quatity	Sensor	Typical devices
Heat	Thermal probe	LM35, DHT11, DS18B20
Light	Light-sensitive transistor	LDR-5516, Light detectors
Sound	Microphones	Sound Sensor KY-038
Distance	Ultrasonic distance measurer	HC-SR04
Touch	Capacitive touch plate	Touch switches
Movement	Infra-red movement detector	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



```
const int LED = 2;

void setup()
{
  Serial.begin(115200);
  pinMode(LED, OUTPUT);
}

void loop()
{
  int value = analogRead(A0);
  Serial.println(value);
  if (value > 512)
    digitalWrite(LED, HIGH);
  else
    digitalWrite(LED, LOW);
}
```



Turns ON if value > 512

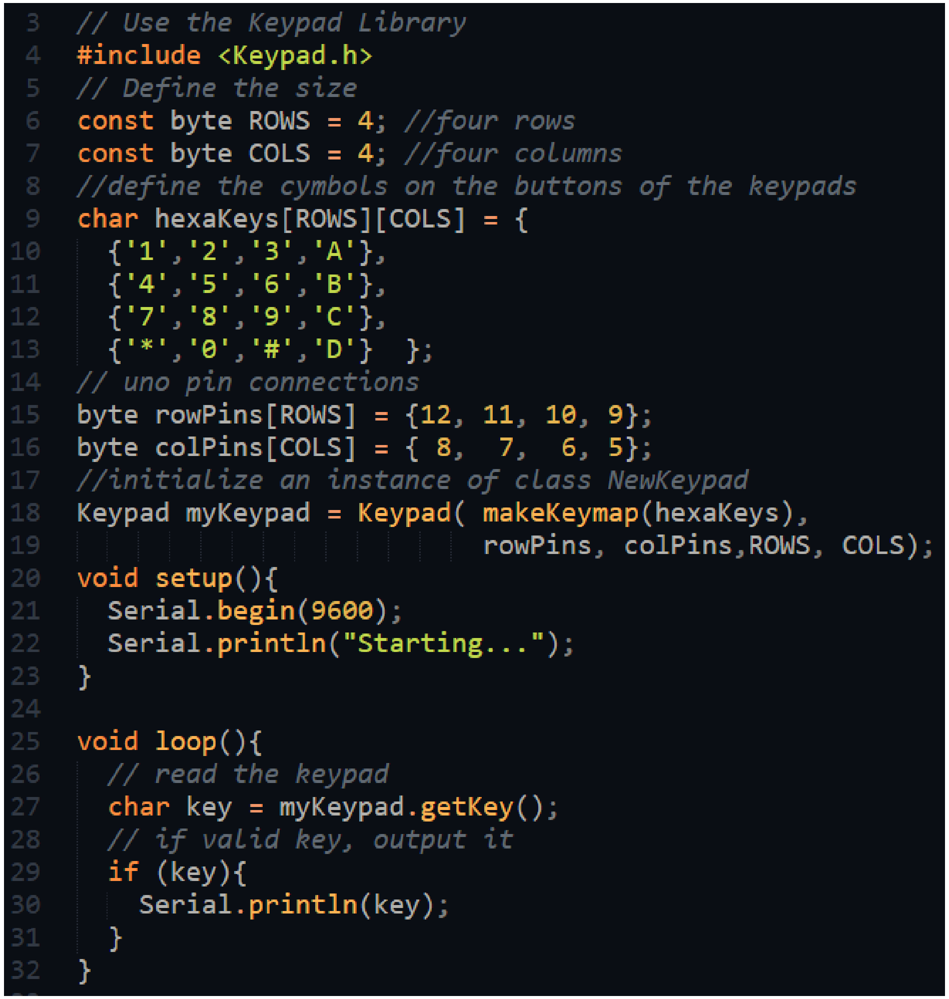
R depends on type of LDR

Light-depended Resistor (LDR 5516)

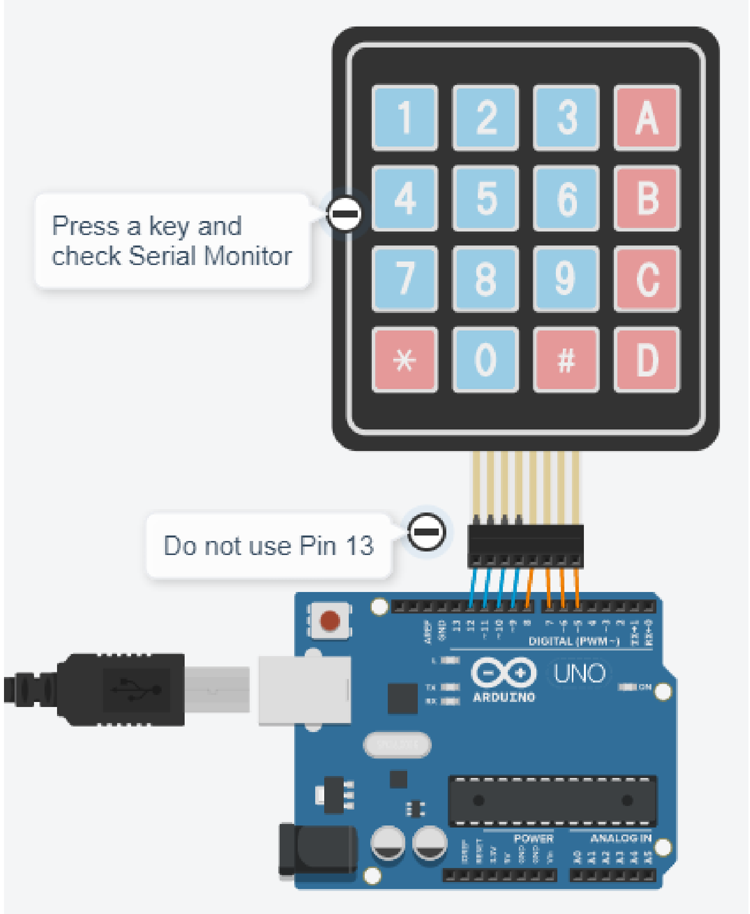
Read the equivalent analog voltage

Matrix Keypad with Library

KeyPad Library



```
// Use the Keypad Library
#include <Keypad.h>
// Define the size
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
//define the cymbols on the buttons of the keypads
char hexaKeys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}  };
// uno pin connections
byte rowPins[ROWS] = {12, 11, 10, 9};
byte colPins[COLS] = { 8,  7,  6, 5};
//initialize an instance of class NewKeypad
Keypad myKeypad = Keypad( makeKeymap(hexaKeys),
                          rowPins, colPins,ROWS, COLS);
void setup(){
  Serial.begin(9600);
  Serial.println("Starting...");
}

void loop(){
  // read the keypad
  char key = myKeypad.getKey();
  // if valid key, output it
  if (key){
    Serial.println(key);
  }
}
```



Press a key and check Serial Monitor

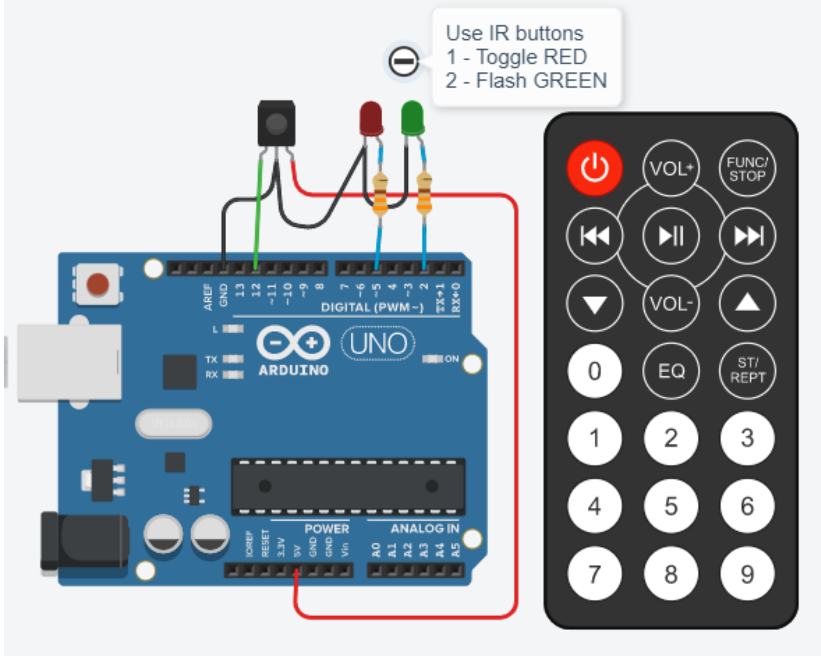
Do not use Pin 13

Simulation only works with some libraries Ref: Libraries with TinkerCAD

Infra Red Remote Control

- Alternative method of providing input to a project.
- The Remote control sends Infra Red (IR) pulses which carry a code. An IR receiver reads the pulses and sends it to the controller input pin Microcontroller decodes the pulses
- Library: Arduino IRRemote
  Simple and can be used with a variety of commercial remotes
- Simplest is to use it with the IRReceiver and Control for Arduino Kits
- A very good YouTube tutorial: DroneBot Workshop: Using IR Remote Controls with Arduino
- Can be simulated with TinkerCAD

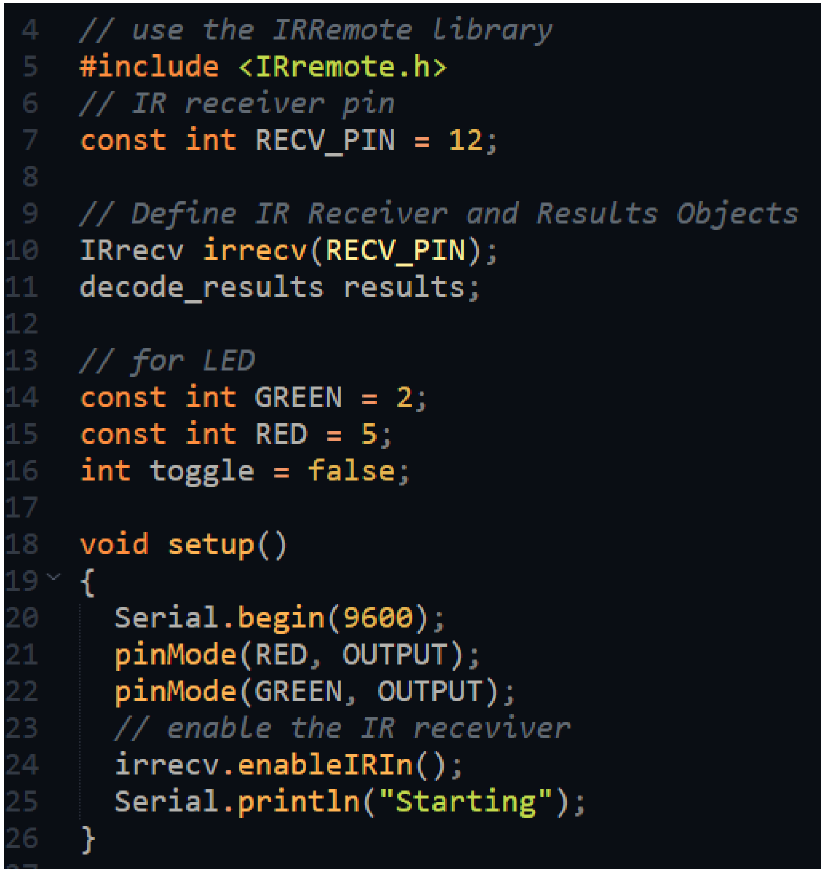
IRRemote Example



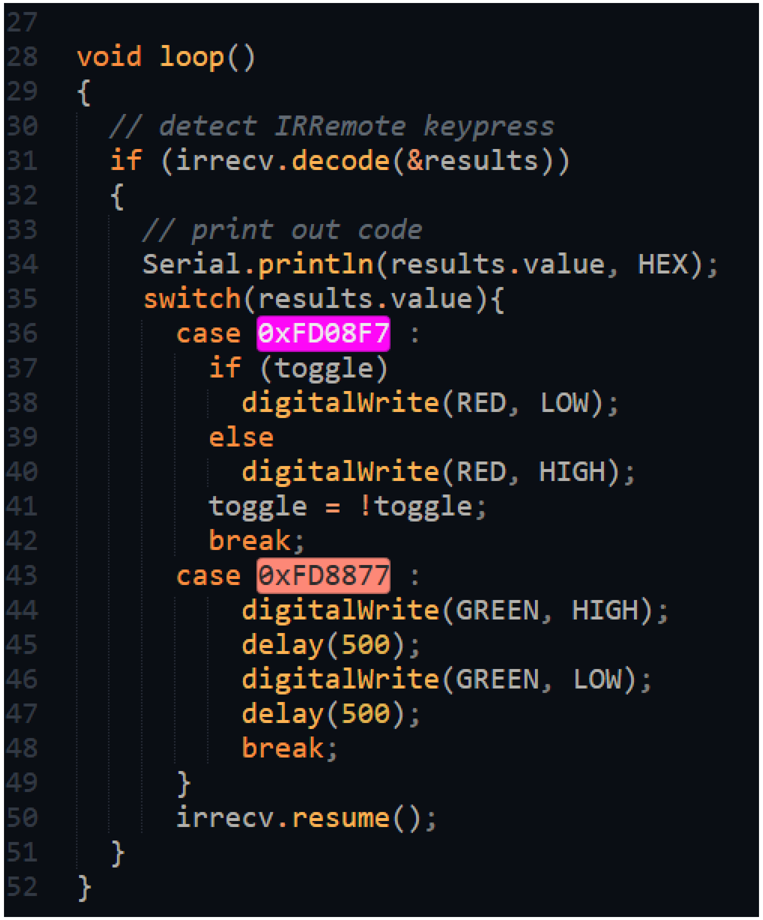
Use IR buttons
1 - Toggle RED
2 - Flash GREEN

- Uno IRRemote Control
- Use Serial Monitor to determine the hex codes before writing the application for the IR Remote
- Example:
  - Key 1 = 0xFD08F7
  - Key 2 = 0xFD8877
- Use a switch-case to effect the applications to be done

IR Remote code



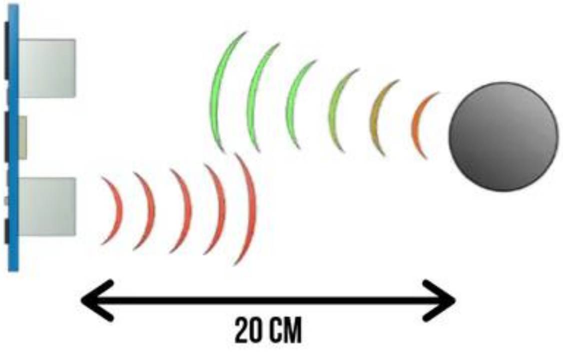
```
// use the IRRemote library
#include <IRremote.h>
// IR receiver pin
const int RECV_PIN = 12;

// Define IR Receiver and Results Objects
IRrecv irrecv(RECV_PIN);
decode_results results;

// for LED
const int GREEN = 2;
const int RED = 5;
int toggle = false;

void setup()
{
  Serial.begin(9600);
  pinMode(RED, OUTPUT);
  pinMode(GREEN, OUTPUT);
  // enable the IR receviver
  irrecv.enableIRIn();
  Serial.println("Starting");
}
```



```

void loop()
{
  // detect IRRemote keypress
  if (irrecv.decode(&results))
  {
    // print out code
    Serial.println(results.value, HEX);
    switch(results.value){
      case 0xFD08F7 :
        if (toggle)
          digitalWrite(RED, LOW);
        else
          digitalWrite(RED, HIGH);
        toggle = !toggle;
        break;
      case 0xFD8877 :
          digitalWrite(GREEN, HIGH);
          delay(500);
          digitalWrite(GREEN, LOW);
          delay(500);
          break;
      }
      irrecv.resume();
  }
}
```

Measuring Distance

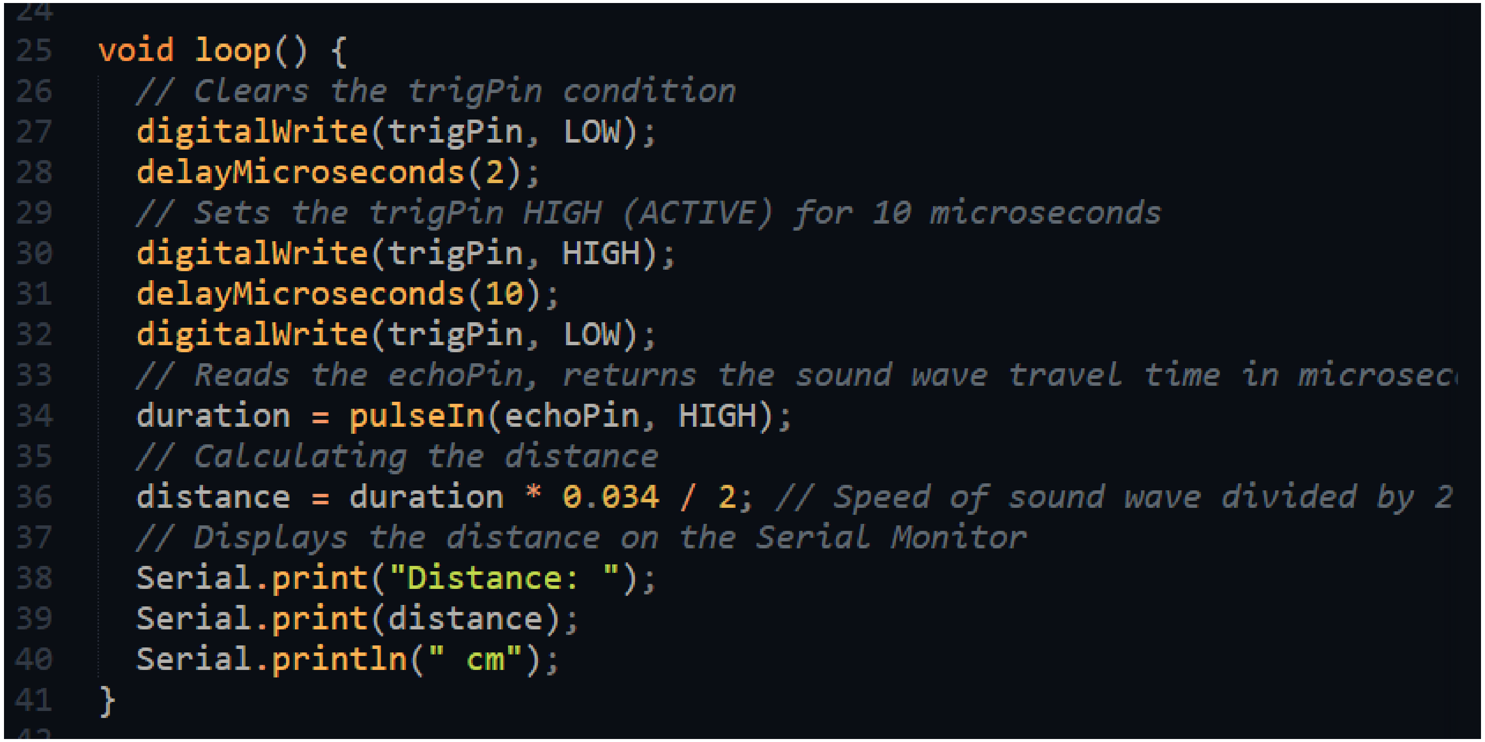
- Ultrasonic Sensor HC-SR04
- Measures distances between 2cm to 400cm without contact using sound (ultrasonic)
- Requires a trigger and an input (2 digital pins)
- Lots of tutorials:
  - Arduino.cc
  - Instructables



SPEED OF SOUND:
v = 340 m/s
v = 0.034 m/s

TIME = DISTANCE/SPEED
t = s/v = 20/0.034
= 588 us
s = t x 0.034/2

Measuring Distance – SR04

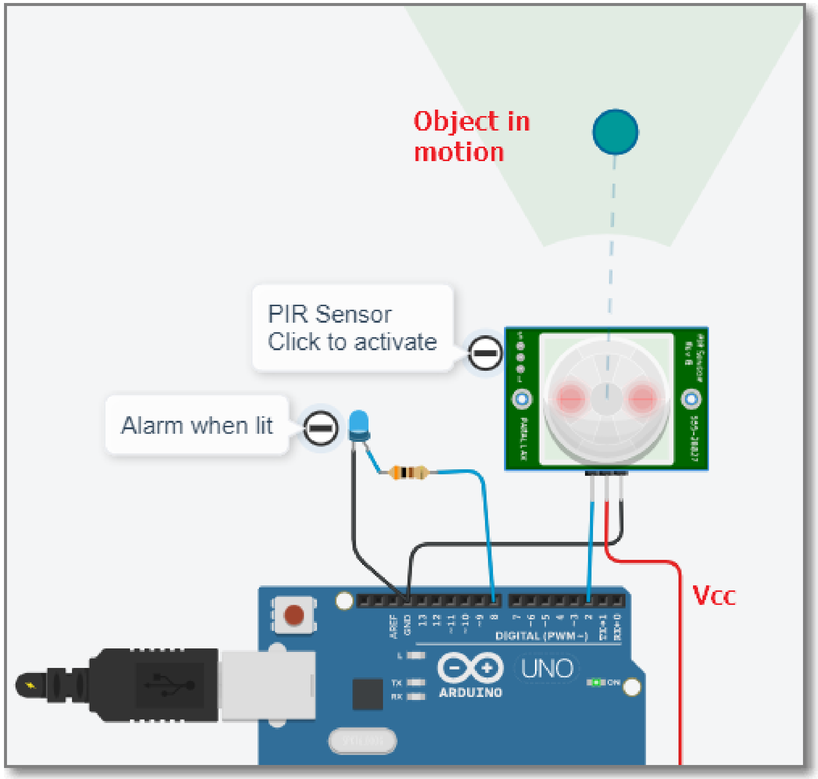


```
void loop() {
  // Clears the trigPin condition
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microsec
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2
  // Displays the distance on the Serial Monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");
}
```

- 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

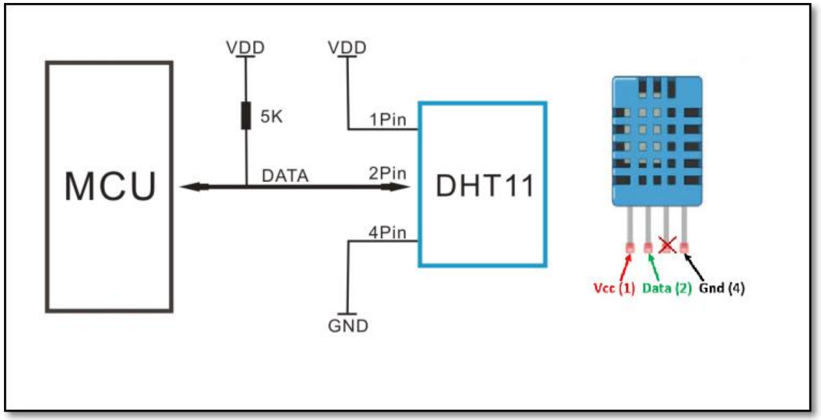
- Passive Infra Red
- Detects motion
  - Adjust Sensitivity
  - Wait at least 15 s
- No Library required, 1 digital I/O input pin for status.
- Check pin
  - LOW no motion
  - HIGH motion detected
- Cannot measure distance



Motion detection using PIR HC-SR501 Better alternative: RCWL-0516 Microwave Proximity Sensor

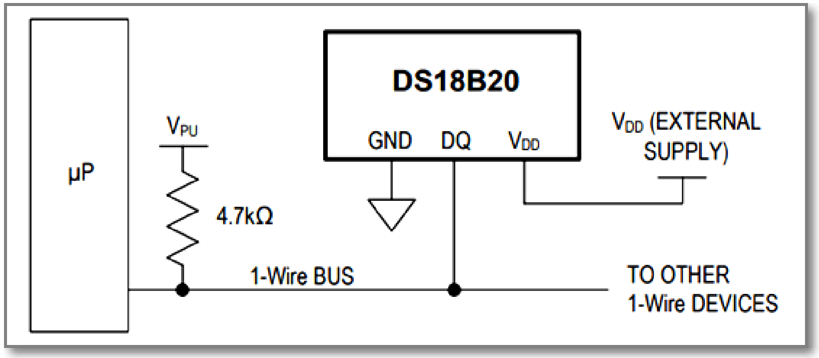
Room Temperature & Humidity

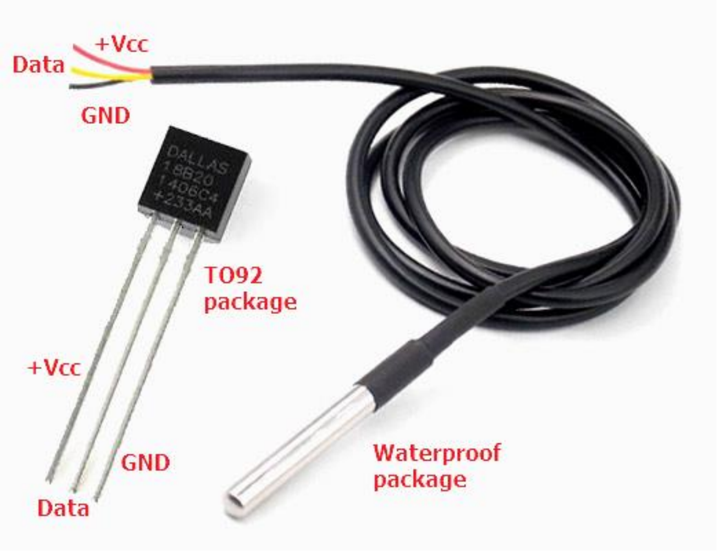
- DHT-11 Temperature and Humidity sensor
- 20~80% humidity, 0~50°C
  1 Hz sampling rate.
- Library from Adafruit (install both):
  - TinyDHT
  - TinyWire
- Requires 1 digital I/O pin
- Better results, accuracy with the DHT-22, however, 2~3x more expensive



Higher Temperatures: DS18B20

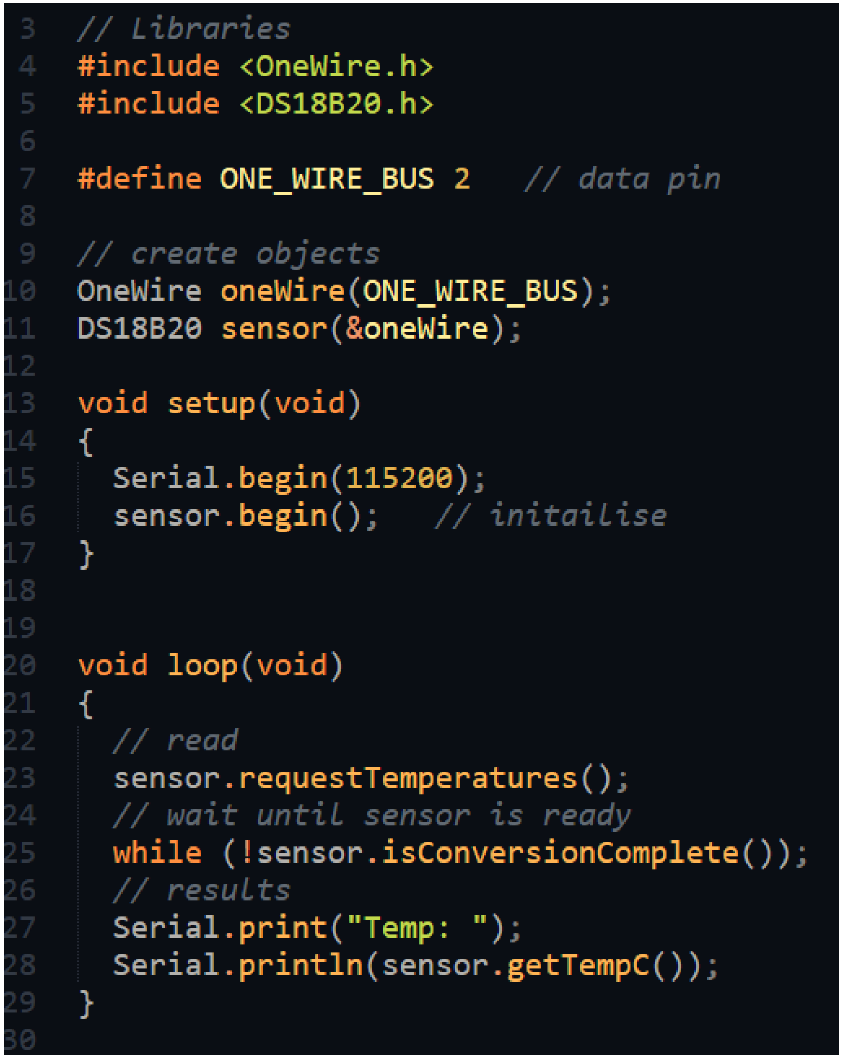
- Specifications:
  - Temperature range:
    -55°C ~ 125°C, Accuracy: +/-0.5°C
  - Communication: 1Wire
  - Sampling: 750mS at 12bit
- Library: DS18B20_RT Arduino Temperature Control Library
  - Minimal functions, simple
  - 1 sensor per MCUpin
- Uses: temperature sensing in hard environments, liquids away from processing unit





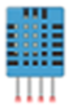
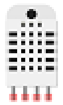
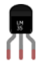
Reading the DS18B20

- Libraries:
  - DS18B20_RT sensor library
  - OneWireNG communications library
- Library only provides minimal functions
  - Instantiate object
  - Trigger sensor read
  - check data ready
  - Read temperature ^{o}C



```
// Libraries
#include <OneWire.h>
#include <DS18B20.h>

#define ONE_WIRE_BUS 2   // data pin

// create objects
OneWire oneWire(ONE_WIRE_BUS);
DS18B20 sensor(&oneWire);

void setup(void)
{
  Serial.begin(115200);
  sensor.begin();   // initailise
}


void loop(void)
{
  // read
  sensor.requestTemperatures();
  // wait until sensor is ready
  while (!sensor.isConversionComplete());
  // results
  Serial.print("Temp: ");
  Serial.println(sensor.getTempC());
}

```

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

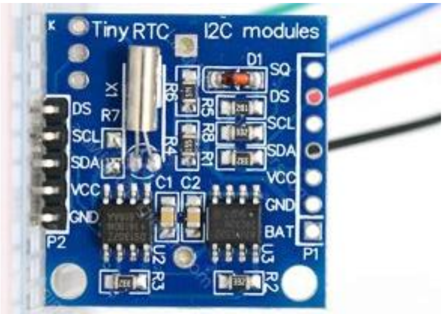
Real Time Clock Modules

- 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

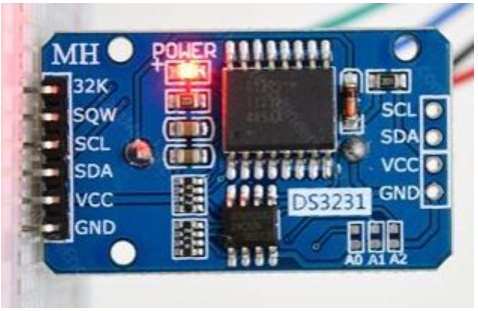
Real Time Clock Modules

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

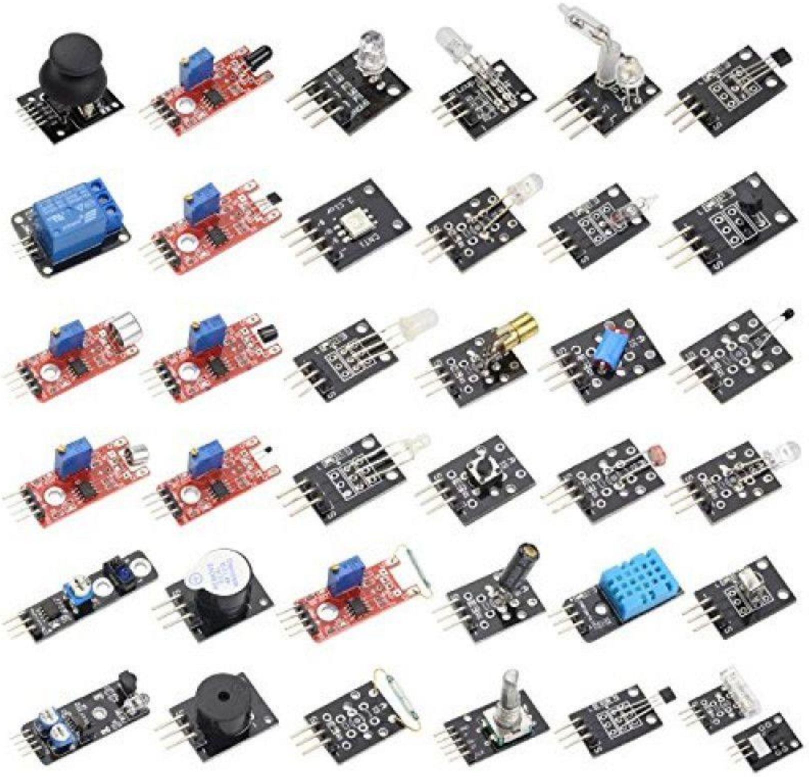
- DS1307
  - Uses OneWire communications
  - Has a 32KHz crystal oscillator
  - Has provision for DS18B20 connection
  - Has battery backup



- DS3231
  - Uses I2C communications
  - Uses temperature controlled oscillator
  - Has battery backup



Sensor Kit 37-in-1



- Almost all physical properties can be measured.
- Affordable way of learning how to work with sensors.
- Code & Tutorials available
- Libraries and simplicity make the Arduino system popular.

Not all are sensors, some are actuators

References:

- Dronebot workshop
- Last Minute Engineers
- Arduino Project Hub
- Instructables

EP1000

Sensors

End