

# Internet-of-Things Communications

Lasse Berntzen

University of South-Eastern Norway

```
for object to mirror_
mirror_mod.mirror_object

operation == "MIRROR_X":
mirror_mod.use_x = True
mirror_mod.use_y = False
mirror_mod.use_z = False
operation == "MIRROR_Y":
mirror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
operation == "MIRROR_Z":
mirror_mod.use_x = False
mirror_mod.use_y = False
mirror_mod.use_z = True

#selection at the end -add
mirror_ob.select= 1
mirror_ob.select=1
context.scene.objects.active
("Selected" + str(modifier
mirror_ob.select = 0
= bpy.context.selected_object
data.objects[one.name].select

print("please select exactly

-- OPERATOR CLASSES -----

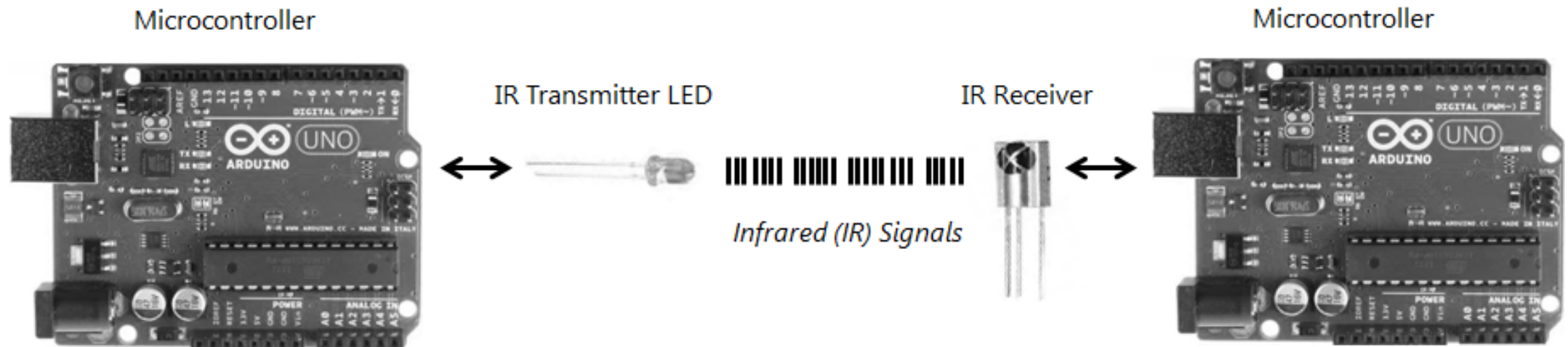
types.Operator):
X mirror to the selected
object.mirror_mirror_x"
mirror X"
```

# Prototyping with Arduino – Chapter #9

- Understanding infrared communications
- Hacking into an existing remote control
- Building an Infrared receiver device
- Using IR receiver TSOP1738/TSOP1838
- Using IR receiver SM0038
- Building an Infrared transmitter device
- Using IR transmitter LED
- Controlling Arduino projects

# Understanding Infrared Communications

- **Infrared light**, Visible light, Ultraviolet light
- A transmitting device (transmitter LED + microcontroller)
- A receiving device (receiver unit + microcontroller)



# Infrared Communication Frequency

- The most common frequency to transmit IR signals is 38 KHz (Kilo Hertz), in other words, the IR LED will be triggered 38,000 times in 1 second by the microcontroller.
- In response to the microcontroller pulses, the IR LED will also emit quick and short IR waves 38,000 times in 1 second.
- These 38,000 IR waves will be carried through the air and received by the IR receiver hardware.

# Infrared Communication Protocol



Several infrared communication protocols



**CONSUMER INFRARED (CIR)**



Used by many remote controllers (but not all)



Some remote controllers use radio signals



Different manufacturers has their own protocols



Lack of standardization

# Hacking Into an Existing Remote Control

What we will do in this lecture:

Using IR receiver TSOP Series IR receivers

Building an infrared transmitter device

Transfer data from one Arduino to another Arduino

Transfer data from Arduino to ESP8266

Building an  
Infrared  
Receiver  
Device

One Arduino Uno R3 with  
USB cable

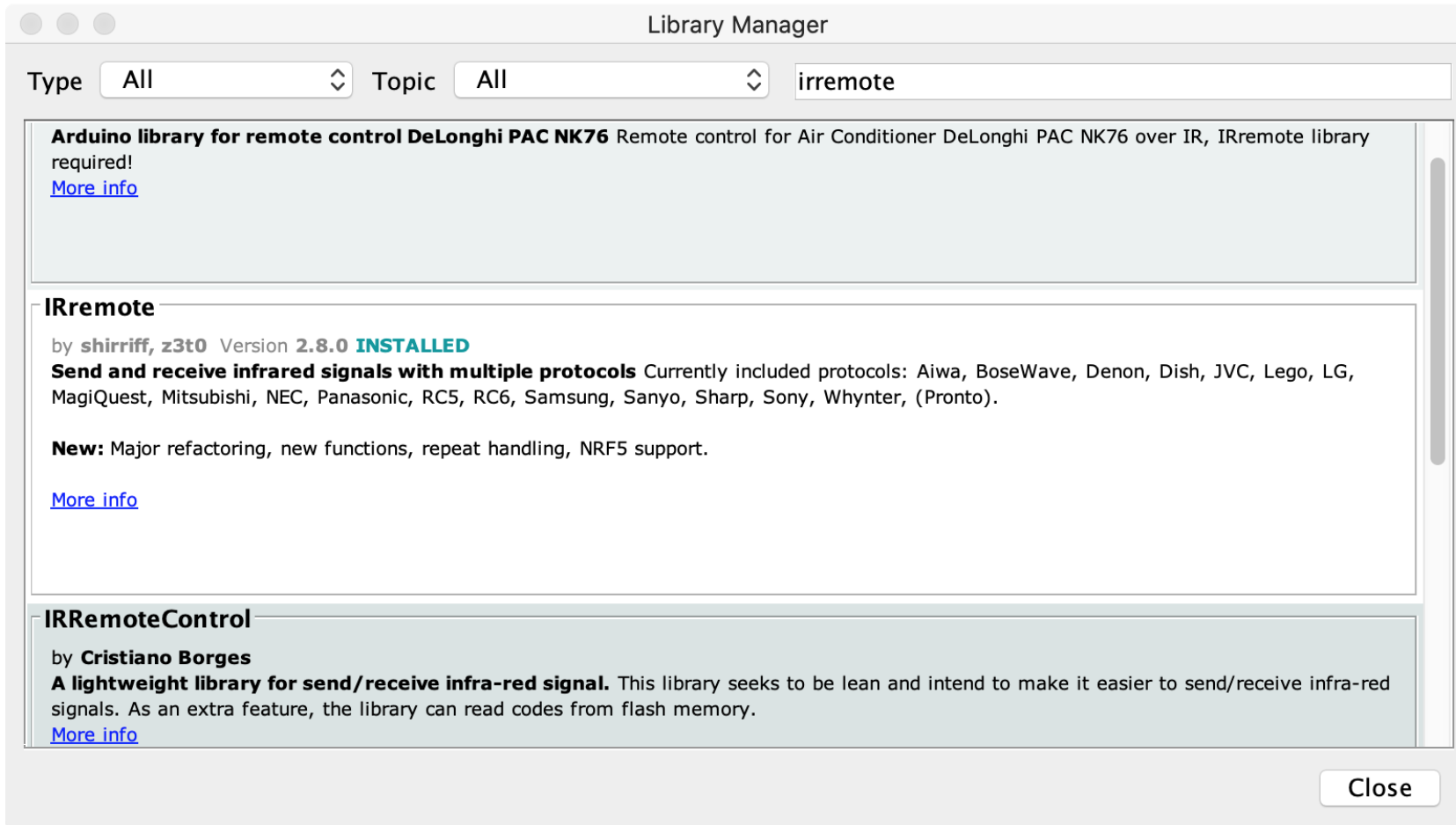
One IR receiver (TSOP family  
of IR receivers) module

Male-to-male jumper wires

# The Arduino Infrared Library

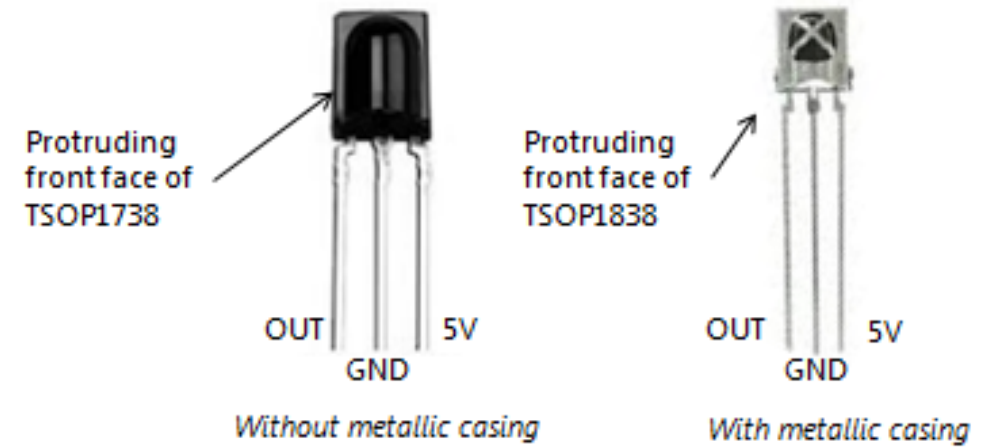
- We need to replace the IRremote library delivered with Arduino
- The built-in library does not support **Consumer InfraRed (CIR)**





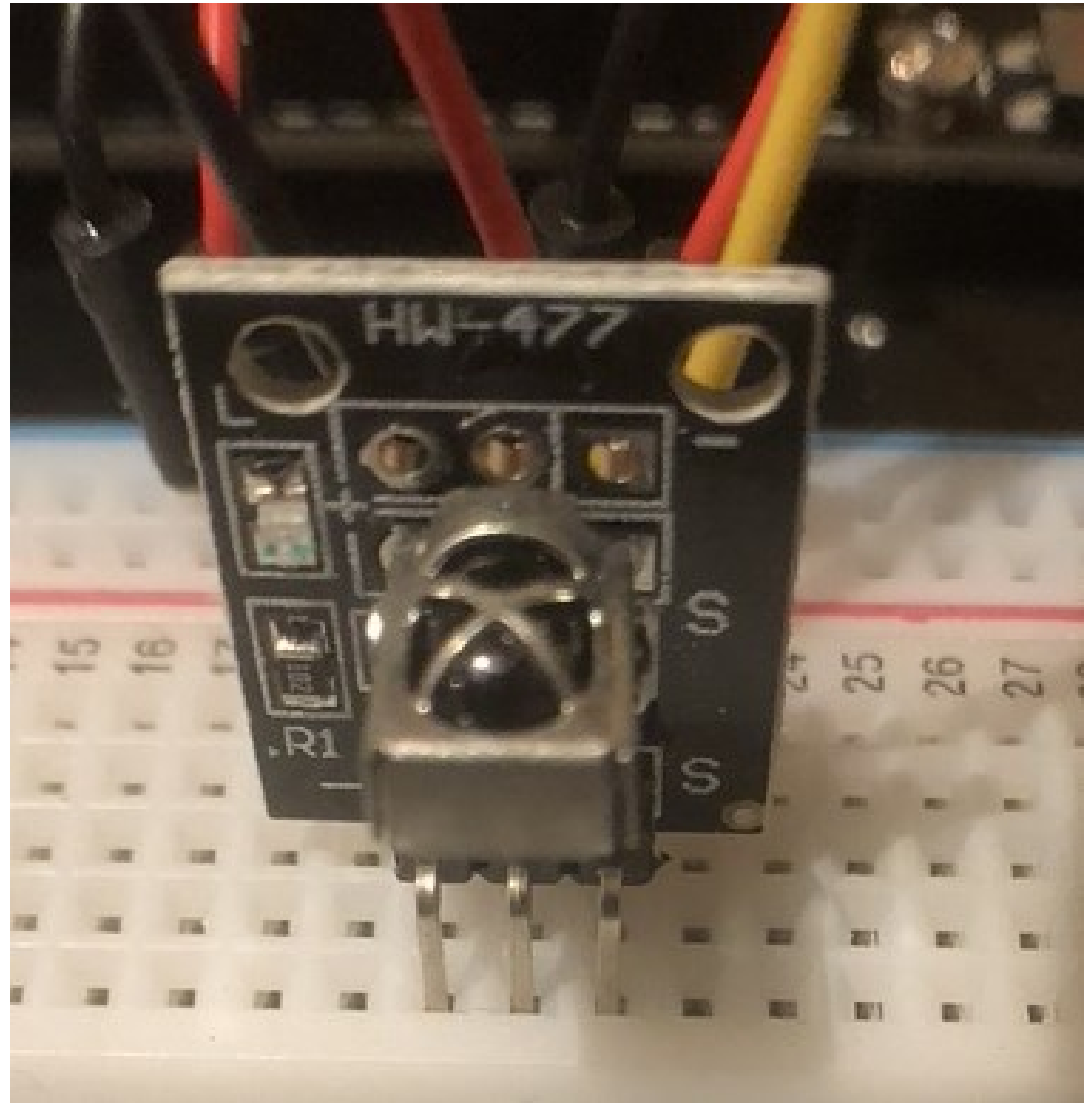
# The Arduino Infrared Library

# Using IR Receiver TSOP Series IR receivers



# The IR Sensor Card

- S = Signal
- - = Gnd
- Middle pin = +5V



# The IR Sensor Card

## Description:

- Infrared receiving module adopts 1838 infrared receiving head  
Light resistance, strong electromagnetic interference, built-in infrared dedicated IC, can work under 500 lux light intensity
- Widely used in: stereo, TV, video machine, disc machine, set-top boxes, digital photo frame, car stereo, remote control toys, satellite receivers, hard disk player, air conditioner, heater, electric fan, lighting and other home appliances

## Specification:

Dimension: 6.4 x 7.4 x 5.1mm

Receiving angle: 90°

Working voltage: 2.7 ~ 5.5V

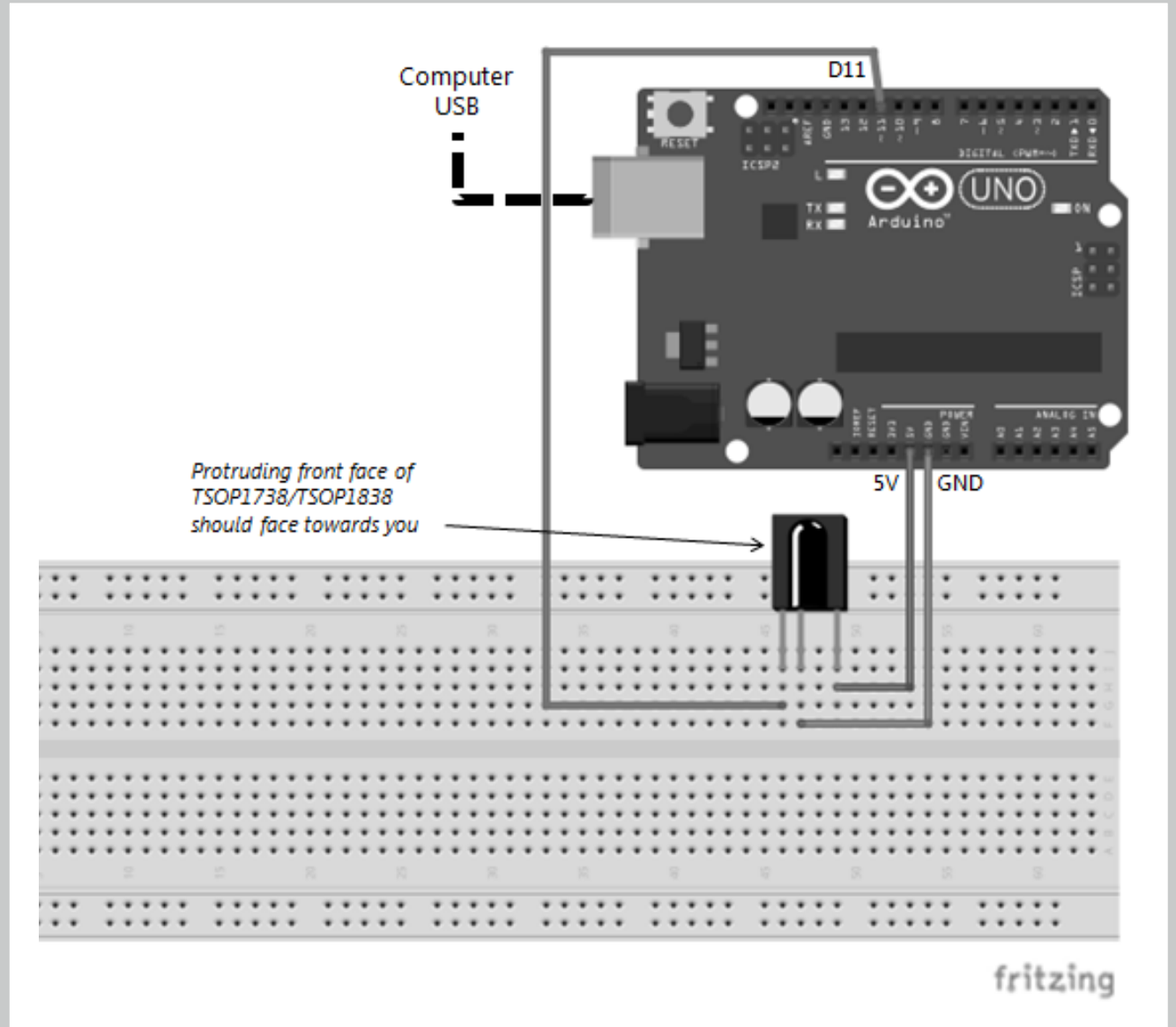
Frequency: 37.9KHz

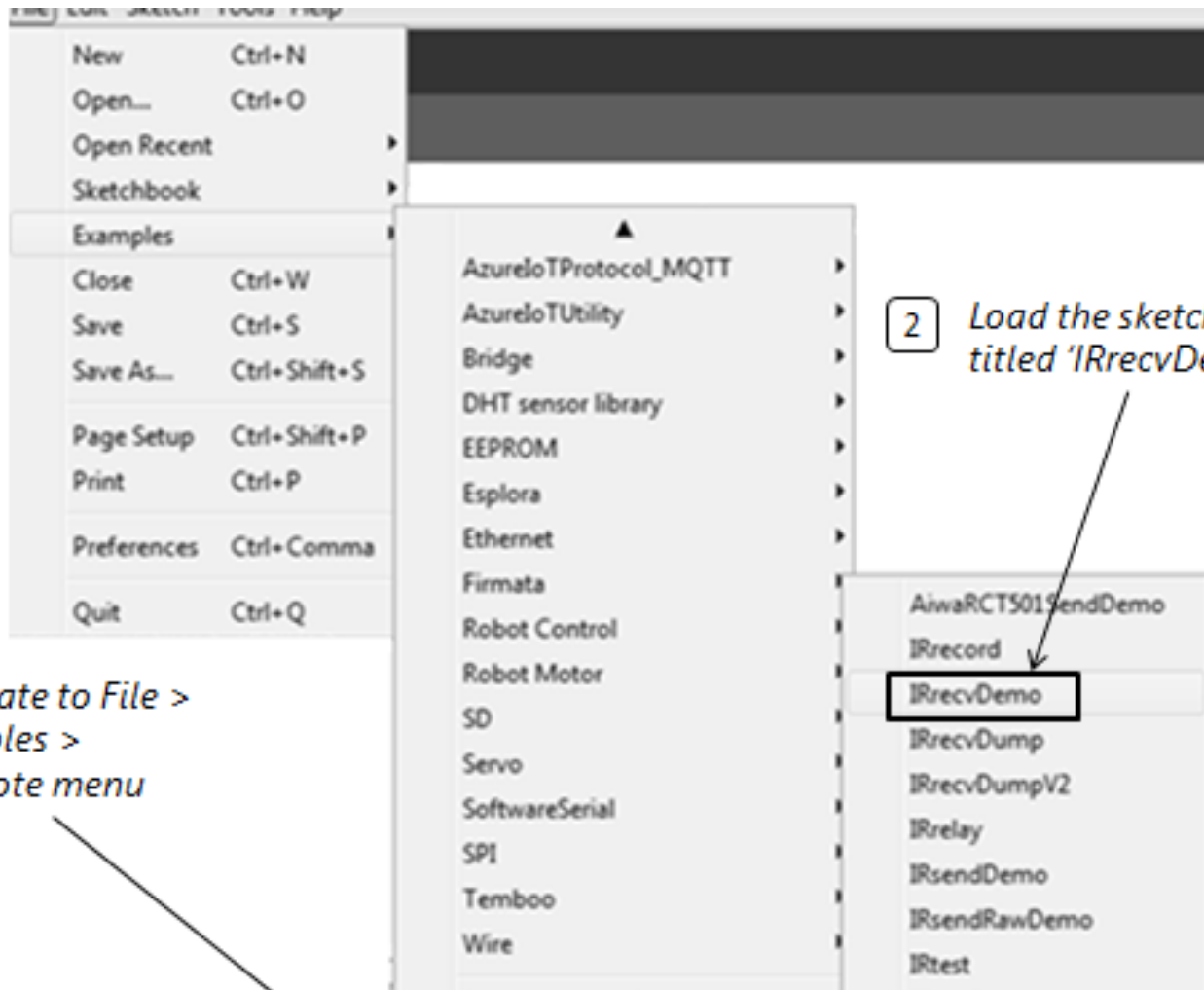
Receiving range: 18m

# The IR Sensor Card

- S: This is the output pin of the IR receiver. This pin is connected to a microcontroller unit. The IR receiver sends the decoded IR signal through this pin. In our case, this pin will be connected to the Arduino board (Pin 11).
- Our Arduino sketch will read the decoded IR signal from this pin.
- GND (-): This pin will be connected to the ground pin of the Arduino board.
- 5V: this pin will be connected to the 5 volt power supply pin of the Arduino.

# Schematic From the Book





1 *Navigate to File > Examples > IRremote menu*

2 *Load the sketch titled 'IRrecvDemo'*

Examples in the Library

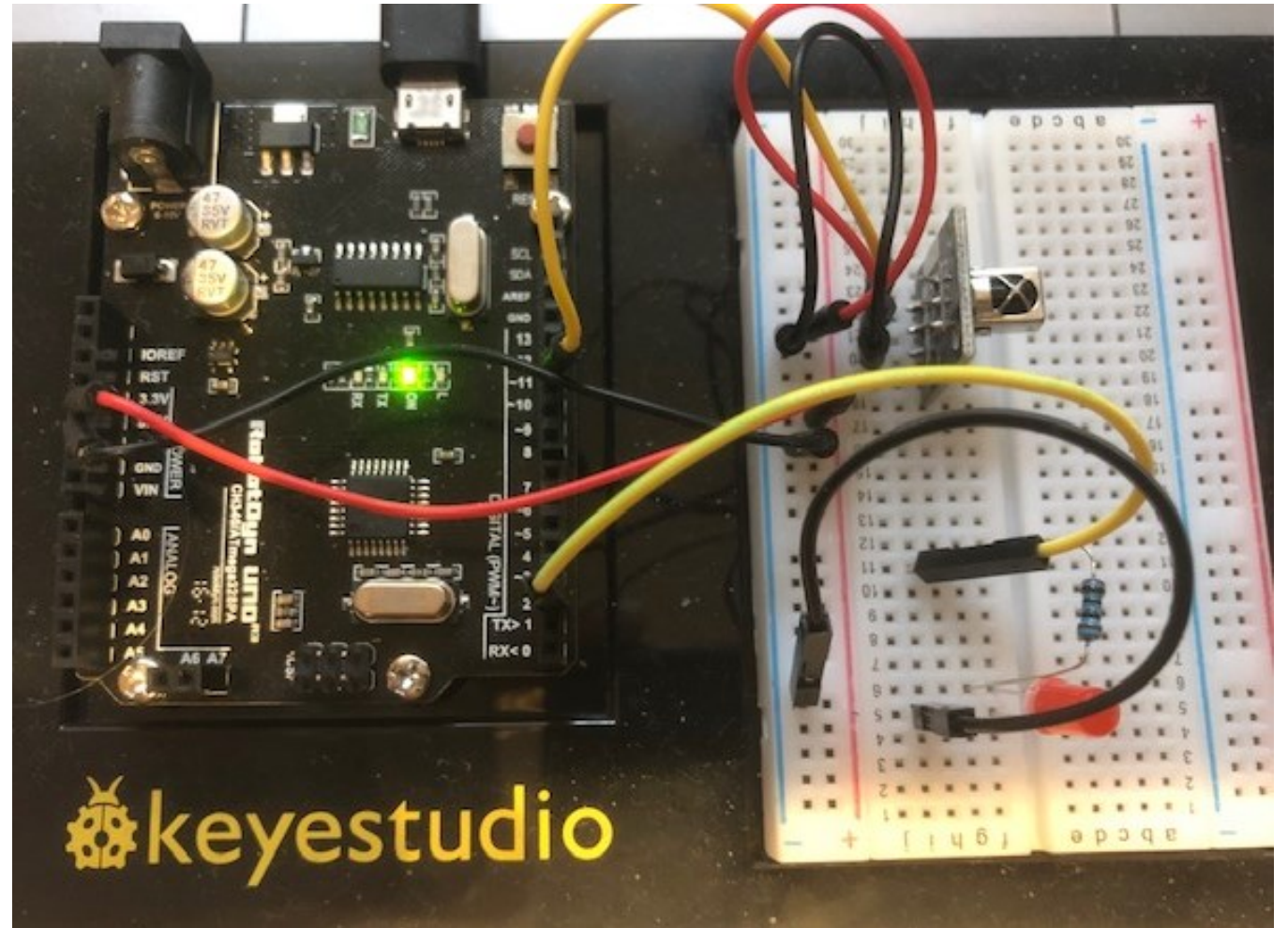
# The circuit

## LED

Cathode to GND (shorter leg)  
Anode to 220 Ohm resistor  
220 Ohm resistor to D2

## IR RECEIVER

S to D11  
Vcc to 5V (middle pin)  
GND to GND





# Sketch (1)

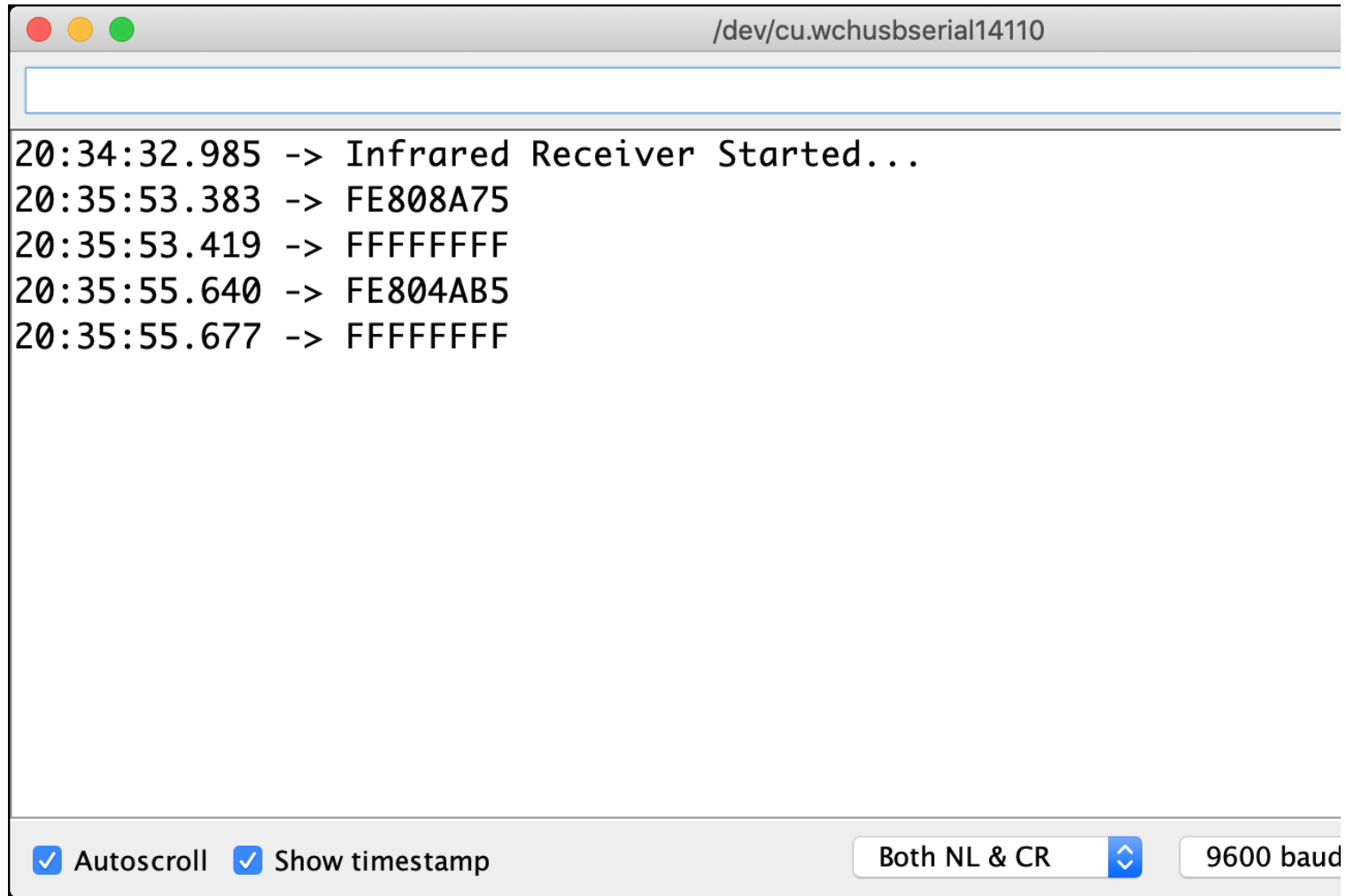
```
// This sketch has been modified based on the original sketch
// written by Ken Shirriff
#include <IRremote.h>
#define RECV_PIN 11 //Pin for IR-receiver
IRrecv irrecv(RECV_PIN); // Create IRrecv object
decode_results results; // Create object to store results
```

# Sketch (2)

```
void setup()
{
  irrecv.enableIRIn();    // Start the receiver
  Serial.begin(9600);     // Start serial communications
  delay(2000);
  Serial.println("Infrared Receiver Started...");
  pinMode(2, OUTPUT);    // Connect LED to GPIO2
  digitalWrite(2, LOW);  // Turn LED off from beginning
}
```

# Sketch (3)

```
void loop()
{
  if (irrecv.decode(&results))
  {
    Serial.println(results.value, HEX);
    if (results.value == 0xFE808A75) {digitalWrite(2, HIGH);}
    if (results.value == 0xFE804AB5) {digitalWrite(2, LOW);}
    irrecv.resume(); // Receive the next value
  }
}
```



The image shows a serial monitor window titled "/dev/cu.wchusbserial14110". The window displays the following output:

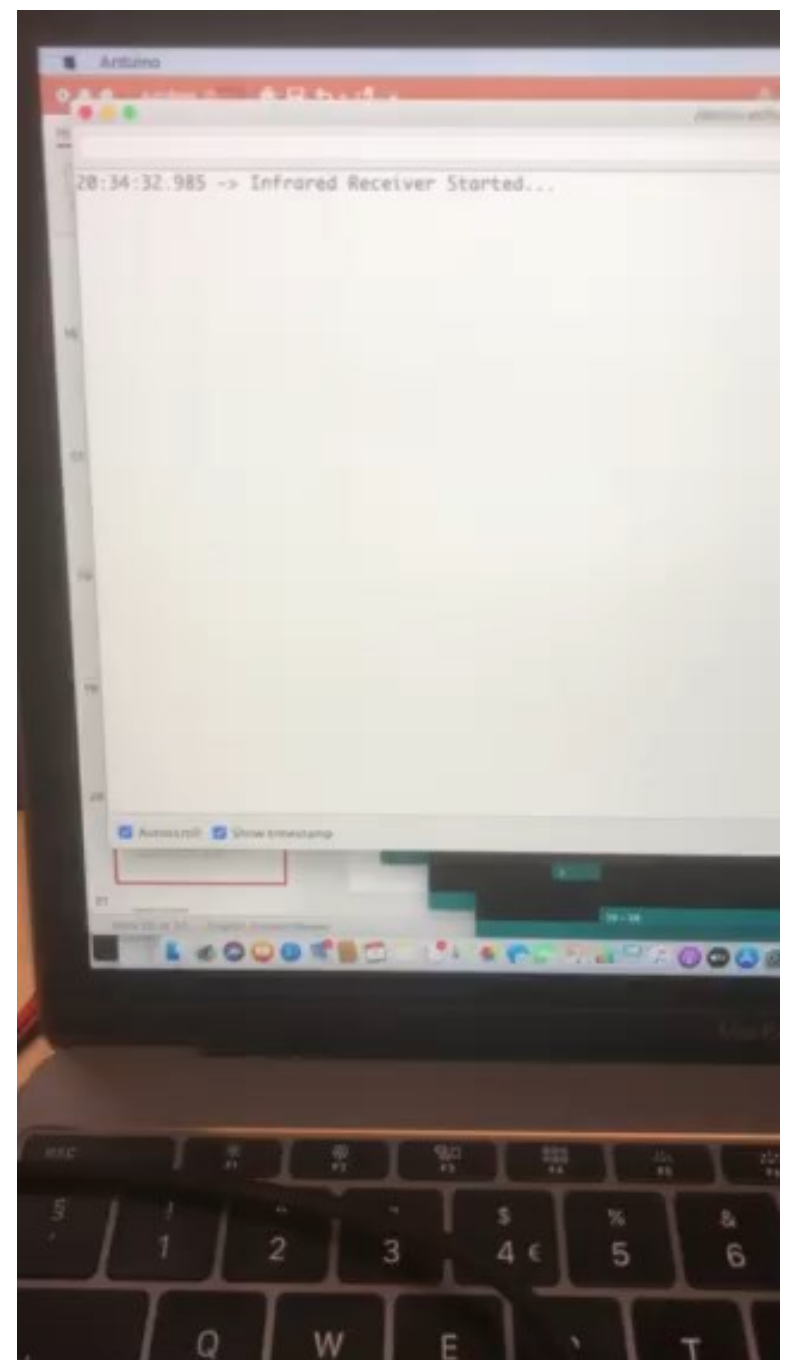
```
20:34:32.985 -> Infrared Receiver Started...
20:35:53.383 -> FE808A75
20:35:53.419 -> FFFFFFFF
20:35:55.640 -> FE804AB5
20:35:55.677 -> FFFFFFFF
```

At the bottom of the window, there are several controls: a checked checkbox for "Autoscroll", a checked checkbox for "Show timestamp", a dropdown menu set to "Both NL & CR", and a text field showing "9600 baud".

Example of  
output from  
serial  
monitor

# Prototype in practical use

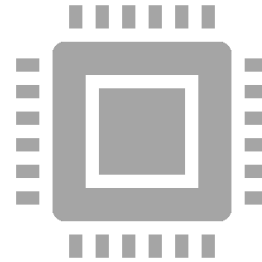
This is a small video demonstration where a remote control is used to turn a LED on and off.



# Detect remote control type



**The library has the ability to detect different kinds of manufacturer controls.**



**Sometimes a manufacturer uses the standard of another manufacturer.**

E.g., my Yamaha receiver is detected as NEC

# Detect remote control type

```
#include "IRremote.h"
IRrecv irrecv(RECV_PIN);
decode_results results;

void setup() {
  Serial.begin(9600);
  irrecv.enableIRIn();
}
```

# Detect remote control type

```
void loop() {  
  if (irrecv.decode(&results)) {  
    switch (results.decode_type) {  
      case NEC: Serial.println("NEC"); break ;  
      case SONY: Serial.println("SONY"); break ;  
      case RC5: Serial.println("RC5"); break ;  
      case RC6: Serial.println("RC6"); break ;  
      case DISH: Serial.println("DISH"); break ;  
      case SHARP: Serial.println("SHARP"); break ;  
      case JVC: Serial.println("JVC"); break ;  
      case SANYO: Serial.println("SANYO"); break ;  
    }  
  }  
}
```



# Detect remote control type

```
    case MITSUBISHI: Serial.println("MITSUBISHI"); break ;
    case SAMSUNG: Serial.println("SAMSUNG"); break ;
    case LG: Serial.println("LG"); break ;
    case WHYENTER: Serial.println("WHYENTER"); break ;
    case AIWA_RC_T501: Serial.println("AIWA_RC_T501"); break ;
    case PANASONIC: Serial.println("PANASONIC"); break ;
    case DENON: Serial.println("DENON"); break ;
default:
    case UNKNOWN: Serial.println("UNKNOWN"); break ; }
    irrecv.resume(); }
}
```

```
Received code  
SONY  
Received code  
NEC  
Received code  
UNKNOWN  
Received code  
UNKNOWN  
Received code  
RC5
```

Autoscroll    Show timestamp   Both NL & CR   9600 baud

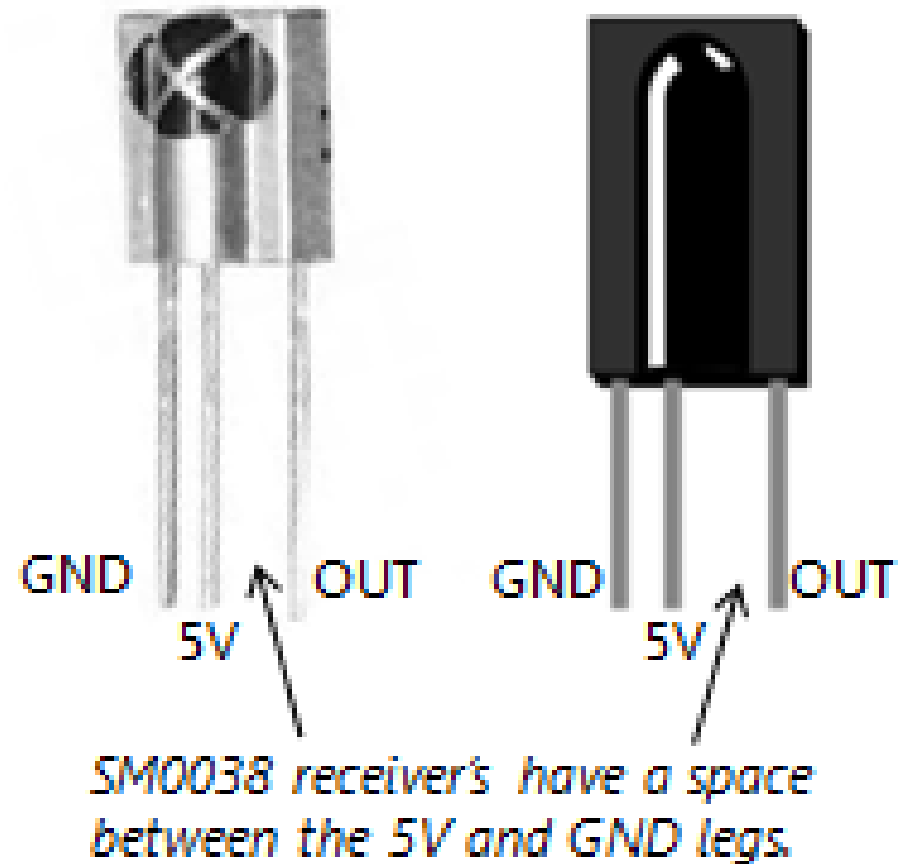
Detect  
remote  
control type

# Specifications

- Supply voltage: 5V
- Working environment: -25~+85°C
- Storage temperature: -30~+100 °C
- Wavelength: 940nm
- Module size: 13.7mm × 27.8mm
- Module weight: 5g
- Signal type: digital signal
- Infrared center wavelength: about 850nm-940nm
- Infrared emission angle: about 20 degrees
- Infrared emission distance: about 1.3 meters (5V 38Khz)

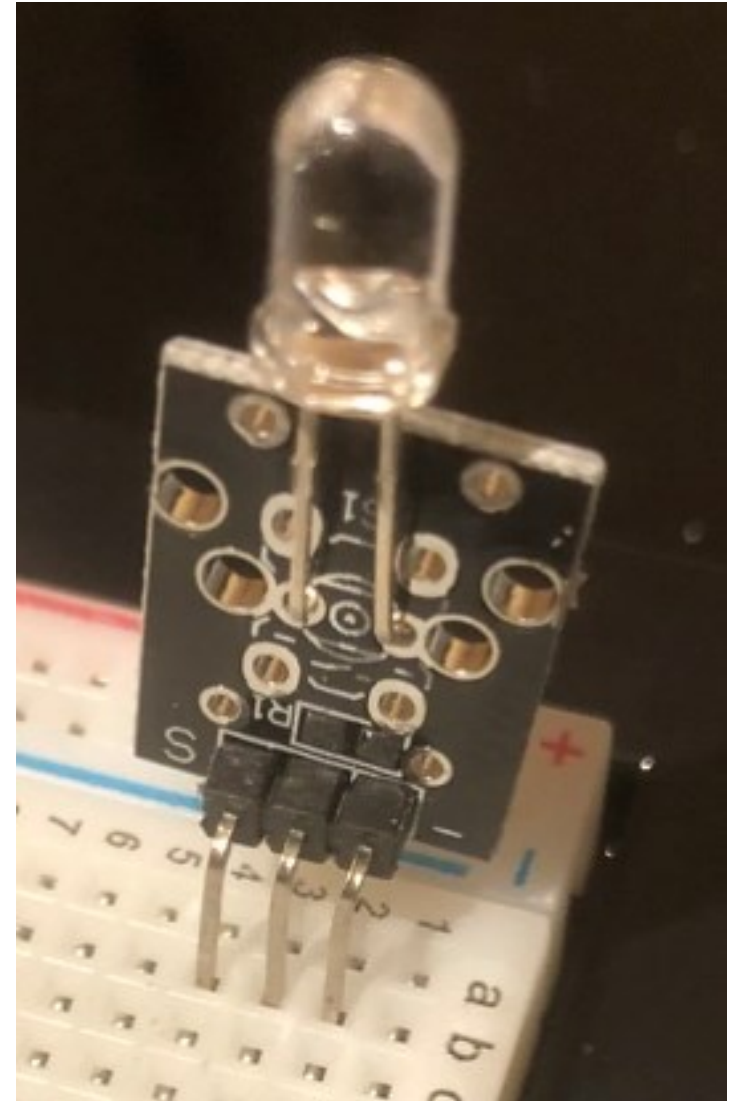
# Using IR Receiver SM0038 (alternative)

- GND: This pin will be connected to the ground pin of the Arduino board.
- 5V: This pin will be connected to the 5 volt power supply pin of the Arduino.
- Out: This is the output pin of the IR receiver. This pin is connected to a microcontroller unit. The IR receiver sends the decoded IR signal through this pin.



# Building an Infrared Transmitter Device

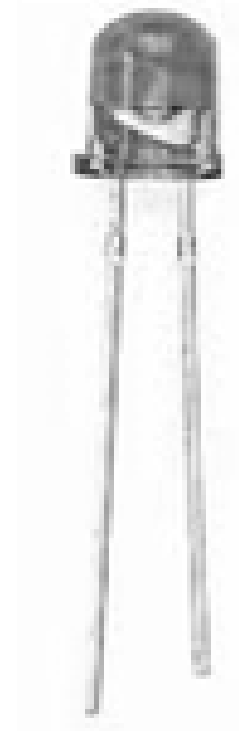
- For transmitting IR signals, we will have to use special IR transmitter LEDs.
- An IR transmitter component is in the sensor set



# Using an IR Transmitter LED

- Positive terminal (longer leg), this pin will be connected to the 5-volt power supply pin of the Arduino.
- Negative terminal (shorter leg), this pin will be connected to the GND pin of the Arduino, via a transistor.

Positive Terminal  
(longer leg)



Negative Terminal  
(shorter leg)

# Components

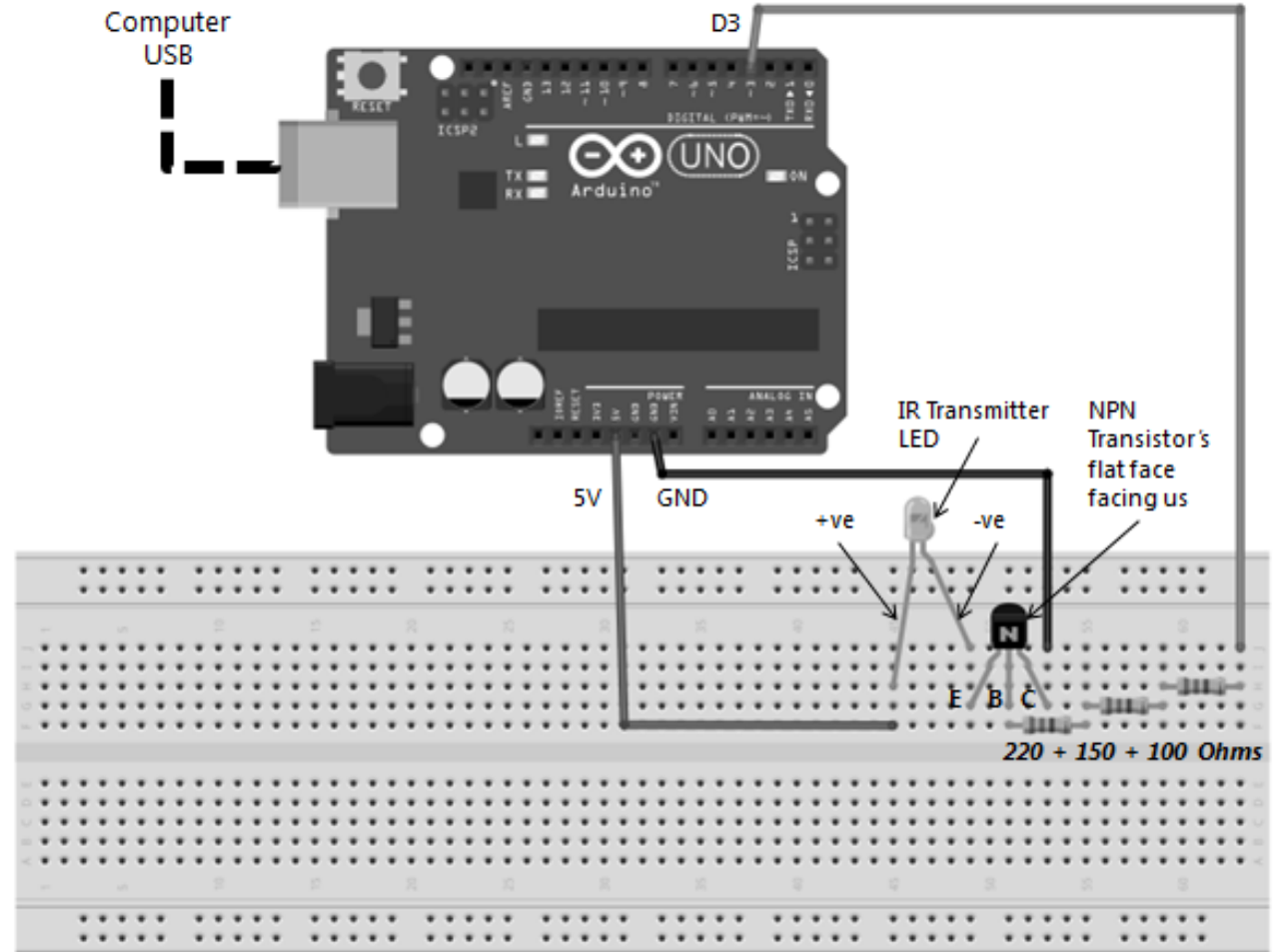
This is different from the textbook, since we use components from the sensor kit.

- Arduino Uno R3 with USB cable
- IR transmitter LED component
- Joystick
- Male-to-male jumper wires

# Schematic

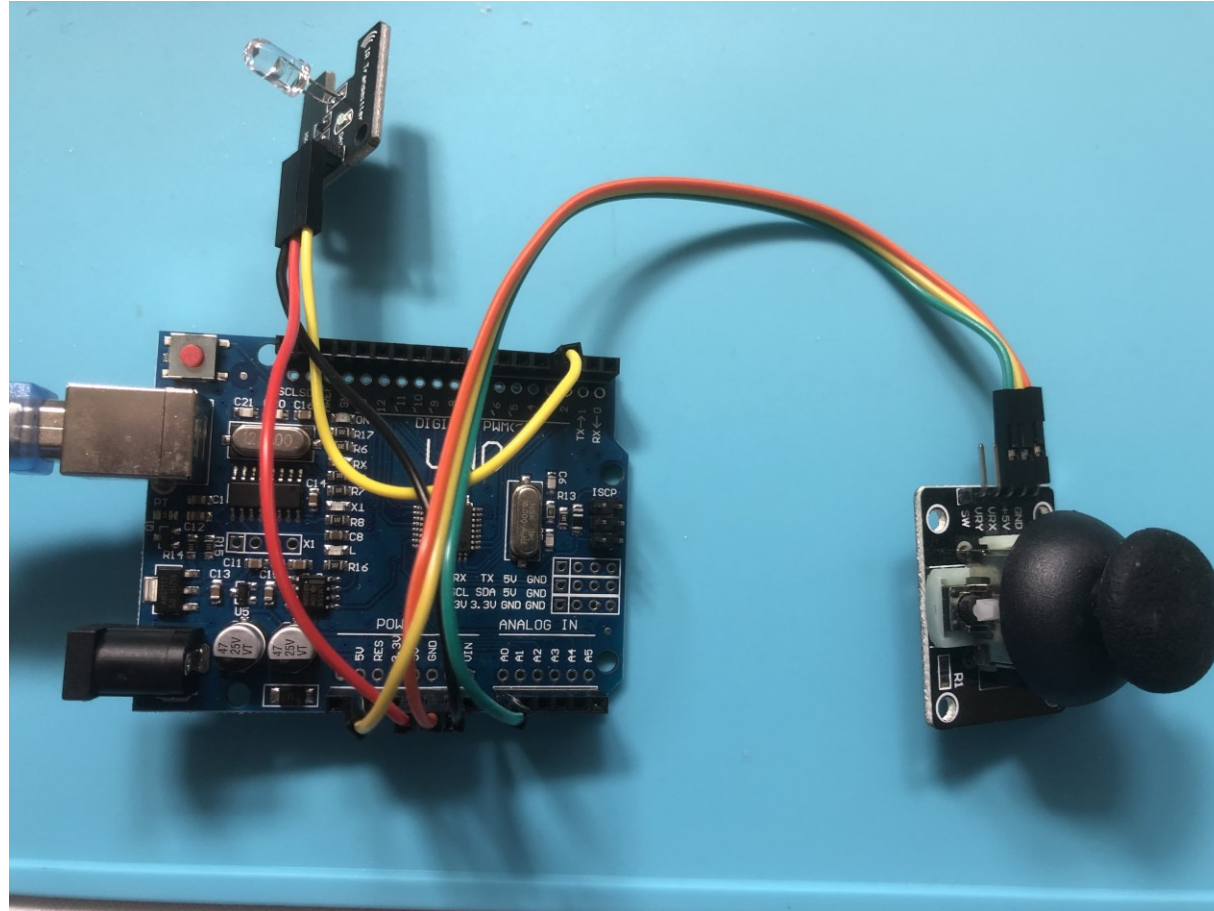
Left pin (S) = Signal, connect to D3  
Middle pin = +5V  
Right pin (-) = GND

We do not need the transistor and the resistors.





# The prototype



# Sketch (1)

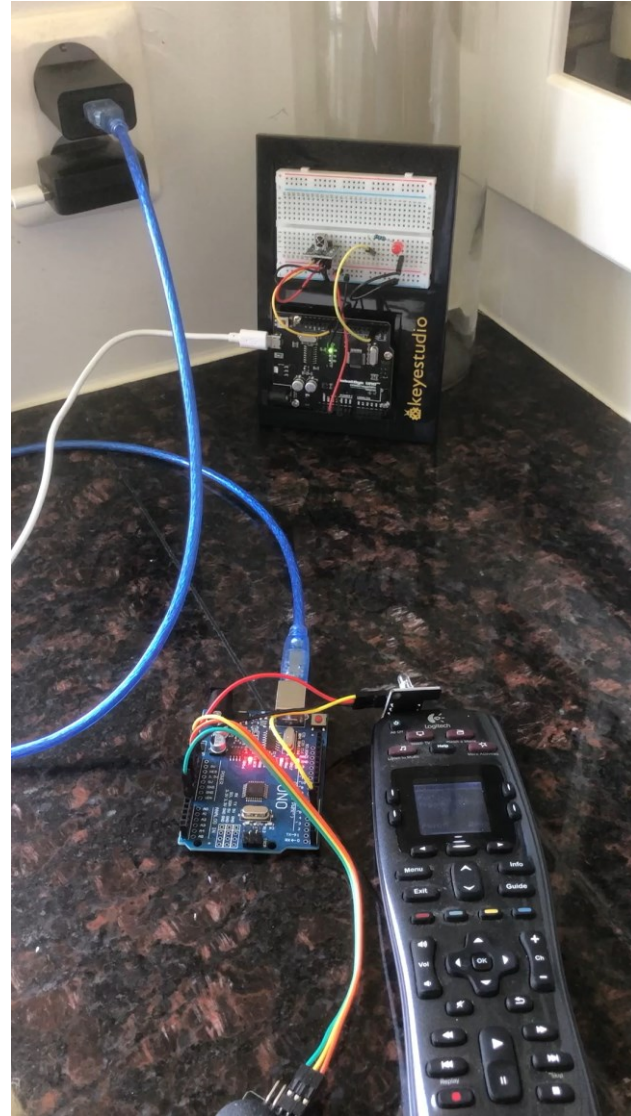
```
#include <IRremote.h>
IRsend My_Sender;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(3000);
  Serial.println("IR sender");
}
```

# Sketch (2)

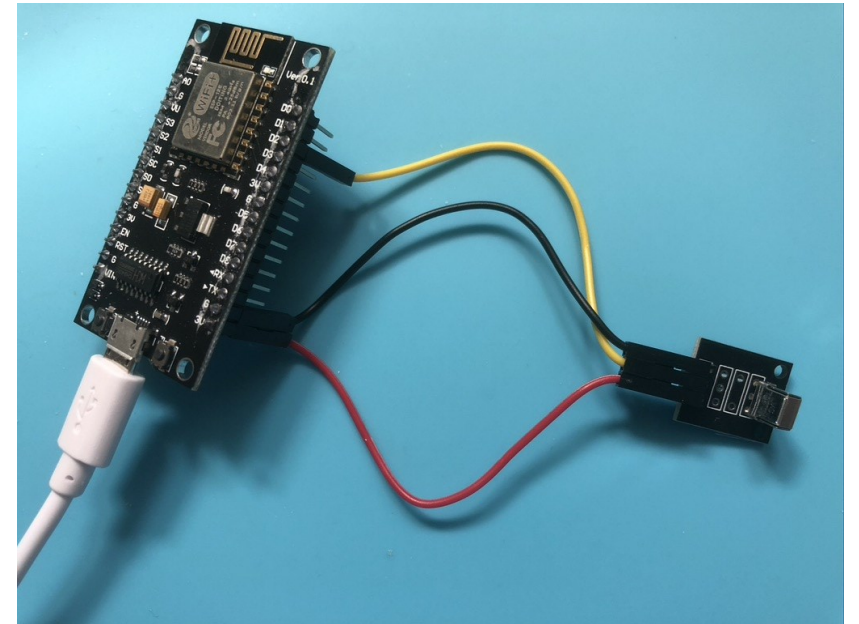
```
void loop() {  
  int i = analogRead(0);  
  Serial.println(String(i));  
  if (i==0) {  
    My_Sender.sendNEC(0xFE808A75, 32);  
    My_Sender.sendNEC(0xFFFFFFFF, 32);  
  }  
  if (i==1023) {  
    My_Sender.sendNEC(0xFE804AB5, 32);  
    My_Sender.sendNEC(0xFFFFFFFF, 32);  
  }  
  delay(50);  
}
```

# Demonstration



# IR receiver on ESP8266

- Different library
  - IRremoteESP8266
- Using built-in LED
  - This is connected to GPIO2 / D4
- IR-receiver
  - This is connected to GPIO4 / D2
- Can be used together with a web-server



# Sketch (1)

```
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#define LED LED_BUILTIN
#define RECV_PIN 4 //Port for IR-receiver
IRrecv irrecv(RECV_PIN); // Create IRrecv object
decode_results results; // Create object to store results
```

# Sketch (2)

```
void setup()
{
  irrecv.enableIRIn(); // Start the receiver
  Serial.begin(9600); // Start serial communications
  delay(2000);
  Serial.println("Infrared Receiver Started...");
  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH); // on ESP8266 this is LED off
}
```

# Sketch (3)

```
void loop()
{
  if (irrecv.decode(&results))
  {
    Serial.println("Result received");
    if (results.value == 0xFE808A75) {digitalWrite(LED, HIGH);}
    if (results.value == 0xFE804AB5) {digitalWrite(LED, LOW);}
    irrecv.resume(); // Receive the next value
  }
}
```



# Demonstration



# Things to Remember

Remember these important points while using the Arduino platform in your future projects.

- The most common frequency to transmit IR signals is 38 KHz (Kilo Hertz).
- The TSOP1738, TSOP1838 and SM0038 are responsive to, and can detect and decode, IR signals of 38 KHz.
- If you want to decode IR signals of a higher frequency than you must use a compatible IR receiver that is responsive to that frequency.
- Use the Arduino IR Library written by Ken Shirriff, to work with generic IR components.

# Things to Remember

- You must remove the pre-existing IR Remote Library before installing the IR Library by Ken Shirriff.
- When using Ken Shirriff's Arduino Library, on an Arduino Uno, only digital I/O Pin 3 can be used for transmitting IR signals. This is because the library internally changes the frequency of pin 3 and utilizes it to send the signals.