



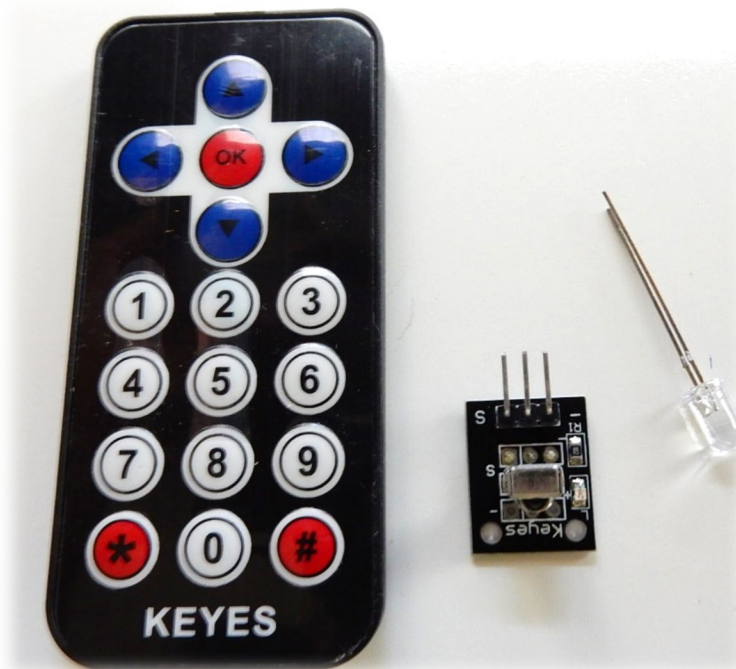
Experiment 4: Infrared Remote Control

Purpose: to install an infrared receiver and program the robot to accept commands from an infrared (IR) remote control to go forward, back, left, right, beep or anything you want to control.

What you will learn:

- Using the IR Remote to control your robot. Decoding IR signals.
- How to work with switch and case statements
- Hexadecimal values

Parts: Shown below is the IR transmitter remote and receiver. There is also an LED looking part. This is an IR transmitter LED that you can use to send IR signals directly from the Arduino. This can be used if you want to create a laser tag game with multiple robots. The LED IR can transmit a signal that can disable other robots for example. Or have the IR LED transmit a signal to a beacon for whatever purpose you want. Your imagination is the limit of what you can do!



Your kit contains an IR remote control and receiver that we will use to remotely control your robot. The idea is that you will be able to send commands with the infrared remote and the Arduino will receive it with the IR receiver sensors to make it move or to turn other parts ON and OFF.

Each time you press a button on the remote you are sending a string of HEX number values that you can then use in your code to make your robot do whatever function you want based on the remote codes.

In mathematics and computing, **hexadecimal** (also base 16, or **hex**) is a positional numeral system with a radix, or base, of 16. It uses sixteen distinct symbols, most often the symbols 0–9 to represent **values** zero to nine, and A, B, C, D, E, F (or alternatively a, b, c, d, e, f) to represent **values** ten to fifteen.

These IR receivers are designed fast flashes light of that is invisible to the human eye. The Arduino is good at triggering these quick flashes thru the IR transmitter LED. The IR remote basically does same thing the Arduino does. When you press a button on the remote there is a chip in the remote that creates these fast pulses that are sent out from the IR LED of the remote. The IR receiver accepts these pulses and code you program selects the pulse and performs the function you programmed it to do. Same thing happens with your TV and remote.

Many TV and other IR sources work between 30 Khz and 45Khz frequency. Different manufacturers will stick to a frequency like Sony likes 44Khz for example on their IR setups.

Some fluorescent lights generate signals in these ranges so it could trigger your robots IR receiver so keep this in mind if your robot starts doing strange things.

Infra means below, so infrared means below red.

The name refers to the fact that the frequency of infrared light waves is less than the frequency of red light waves. The wavelength our IR LED transmits is 980 nanometers (abbreviated nm), and that's the same wavelength our IR receiver detects. This wavelength is in the near-infrared range. The far-infrared range is 2000 to 10,000 nm, and certain wavelengths in this range are used for night-vision goggles and IR temperature sensing.

Library

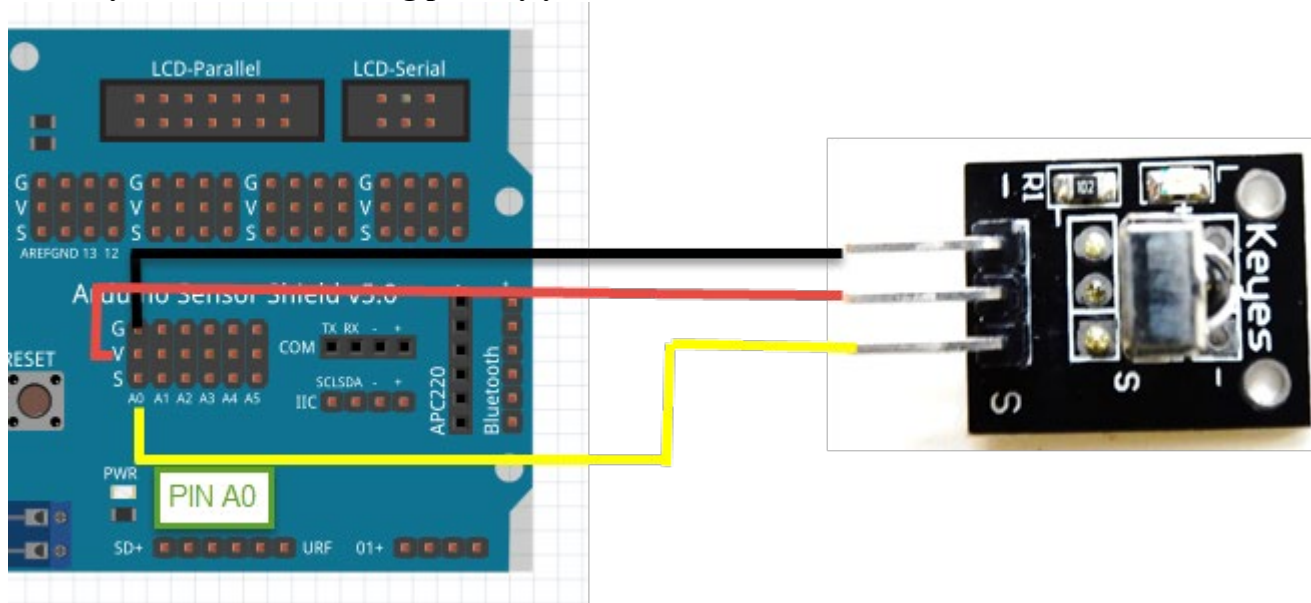
For these experiments you will need to download and copy the IRremote folder to Arduino Library folder under C:\Program Files\Arduino\libraries

For more information on installing libraries check out this reference site:

<https://learn.sparkfun.com/tutorials/installing-an-arduino-library>

Use this sample programs and diagram to get help you test and use the remote.

Note: If you connect the wrong polarity you will burn the sensor!



Testing the remote

```
/*
 * IRremote: IRrecvDemo - demonstrates receiving IR codes with IRrecv
 */

#include <IRremote.h>
int RECV_PIN = A0;
IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    delay(60);
    irrecv.resume(); // Receive the next value
  }
  delay(60);
}
```

Another test program to try.

```
/*
 * IRremote: IRrecvDump - dump details of IR codes with IRrecv
 * An IR detector/demodulator must be connected to the input RECV_PIN.
 * Version 0.1 July, 2009
 * Copyright 2009 Ken Shirriff
 * http://arcfn.com
 * JVC and Panasonic protocol added by Kristian Lauszus (Thanks to zenwheel and other people at the original
 blog post)
 */

#include <IRremote.h>

int RECV_PIN = A0;

IRrecv irrecv(RECV_PIN);

decode_results results;
```

```

void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}

// Dumps out the decode_results structure.
// Call this after IRrecv::decode()
// void * to work around compiler issue
//void dump(void *v) {
// decode_results *results = (decode_results *)v
void dump(decode_results *results) {
  int count = results->rawlen;
  if (results->decode_type == UNKNOWN) {
    Serial.print("Unknown encoding: ");
  }
  else if (results->decode_type == NEC) {
    Serial.print("Decoded NEC: ");
  }
  else if (results->decode_type == SONY) {
    Serial.print("Decoded SONY: ");
  }
  else if (results->decode_type == RC5) {
    Serial.print("Decoded RC5: ");
  }
  else if (results->decode_type == RC6) {
    Serial.print("Decoded RC6: ");
  }
  else if (results->decode_type == PANASONIC) {
    Serial.print("Decoded PANASONIC - Address: ");
    Serial.print(results->panasonicAddress,HEX);
    Serial.print(" Value: ");
  }
  else if (results->decode_type == JVC) {
    Serial.print("Decoded JVC: ");
  }
  Serial.print(results->value, HEX);
  Serial.print(" (");
  Serial.print(results->bits, DEC);
  Serial.println(" bits)");
  Serial.print("Raw (");
  Serial.print(count, DEC);
  Serial.print("): ");
}

```

```

for (int i = 0; i < count; i++) {
  if ((i % 2) == 1) // uses modulo operator
  {
    Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
  }
  else {
    Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
  }
  Serial.print(" ");
}
Serial.println("");
}
void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    dump(&results);
  }
  delay(60);
  irrecv.resume(); // Receive the next value
}
}

```

And more decoding and testing. Optional only if you have a brand name remote to test

```

#include "IRremote.h"

int receiver = A0; // IR receiver to Arduino
IRrecv irrecv(receiver); // create instance of 'irrecv'
decode_results results; // create instance of 'decode_results'

void setup()
{
  Serial.begin(9600);
  Serial.println("IR Receiver Raw Data + Button Decode Test");
  irrecv.enableIRIn(); // Start the receiver
}

void loop()
{
  if (irrecv.decode(&results)) // have we received an IR signal?

  {
// Serial.println(results.value, HEX); Uncomment to see raw values
  translateIR();
  irrecv.resume(); // receive the next value
  delay(60);
}
}

```

```
}  
}
```

void translateIR() // takes action based on IR code received

```
{  
  switch(results.value)  
  {  
    case 0xFFA25D:  
      Serial.println(" CH-      ");  
      break;  
    case 0xFF629D:  
      Serial.println(" CH      ");  
      break;  
    case 0xFFE21D:  
      Serial.println(" CH+      ");  
      break;  
    case 0xFF22DD:  
      Serial.println(" PREV      ");  
      break;  
    case 0xFF02FD:  
      Serial.println(" NEXT      ");  
      break;  
    case 0xFFC23D:  
      Serial.println(" PLAY/PAUSE  ");  
      break;  
    case 0xFFE01F:  
      Serial.println(" VOL-      ");  
      break;  
    case 0xFFA857:  
      Serial.println(" VOL+      ");  
      break;  
    case 0xFF906F:  
      Serial.println(" EQ      ");  
      break;  
    case 0xFF6897:  
      Serial.println(" 0      ");  
      break;  
    case 0xFF9867:  
      Serial.println(" 100+      ");  
      break;  
    case 0xFFB04F:  
      Serial.println(" 200+      ");  
      break;  
    case 0xFF30CF:
```

```

    Serial.println(" 1 ");
    break;
case 0xFF18E7:
    Serial.println(" 2 ");
    break;

case 0xFF7A85:
    Serial.println(" 3 ");
    break;
case 0xFF10EF:
    Serial.println(" 4 ");
    break;
case 0xFF38C7:
    Serial.println(" 5 ");
    break;
case 0xFF5AA5:
    Serial.println(" 6 ");
    break;
case 0xFF42BD:
    Serial.println(" 7 ");
    break;
case 0xFF4AB5:
    Serial.println(" 8 ");
    break;
case 0xFF52AD:
    Serial.println(" 9 ");
    break;
default:
    Serial.println(" other button ");
}
delay(500);
}

```

A neat example of how to turn on a relay or LED with the remote – try it if you like with an LED LED must be connected to pin to 13 for the internal built in LED on the Arduino board.

```

#include <IRremote.h>

int RECV_PIN = A0;

int RELAY_PIN = 13;

IRrecv irrecv(RECV_PIN);

decode_results results;

```

```
// Dumps out the decode_results structure.
// Call this after IRrecv::decode()
// void * to work around compiler issue
//void dump(void *v) {
//  decode_results *results = (decode_results *)v
void dump(decode_results *results) {
    int count = results->rawlen;
    if (results->decode_type == UNKNOWN) {
        Serial.println("Could not decode message");
    }
    else {
        if (results->decode_type == NEC) {
            Serial.print("Decoded NEC: ");
        }
        else if (results->decode_type == SONY) {
            Serial.print("Decoded SONY: ");
        }
        else if (results->decode_type == RC5) {
            Serial.print("Decoded RC5: ");
        }
        else if (results->decode_type == RC6) {
            Serial.print("Decoded RC6: ");
        }
        Serial.print(results->value, HEX);
        Serial.print(" (");
        Serial.print(results->bits, DEC);
        Serial.println(" bits)");
    }
}
```



```
Serial.print("Raw (");
Serial.print(count, DEC);
Serial.print("): ");

for (int i = 0; i < count; i++) {
  if ((i % 2) == 1) {
    Serial.print(results->rawbuf[i]*USECPERTICK, DEC);
  }
  else {
    Serial.print(-(int)results->rawbuf[i]*USECPERTICK, DEC);
  }
  Serial.print(" ");
}
Serial.println("");
}

void setup()
{
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(13, OUTPUT);
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
}

int on = 0;
unsigned long last = millis();

void loop() {
  if (irrecv.decode(&results)) {
```

```

// If it's been at least 1/4 second since the last
// IR received, toggle the relay
if (millis() - last > 250) {
    on = !on;
    digitalWrite(RELAY_PIN, on ? HIGH : LOW);
    digitalWrite(13, on ? HIGH : LOW);
    dump(&results);
}

last = millis();

irrecv.resume(); // Receive the next value
}
}

```

Navigate with the IR remote

When you press a button on the remote the robot goes forward, another button the robot goes backwards, another left, another right, another blink or make sound (in this case is a piece of the Super Mario Bros song – press 1 on the remote to play the song)

/* Use the IR remote to move robot forward, back, left, right and play a sound or song when pressing the number 1 */

```

#include <IRremote.h>
#include <Servo.h>
#include "pitches.h"

const int R_PIN = A0;
const int servoLeftPin = 11;
const int servoRightPin = 10;
IRrecv irrecv(R_PIN);
decode_results result;

Servo servoLeft;
Servo servoRight;

const int LED = 13;

#define melodyPin 4
//Mario main theme melody
int melody[] = {

```

```
NOTE_E7, NOTE_E7, 0, NOTE_E7,  
0, NOTE_C7, NOTE_E7, 0,  
NOTE_G7, 0, 0, 0,  
NOTE_G6, 0, 0, 0,
```

```
NOTE_C7, 0, 0, NOTE_G6,  
0, 0, NOTE_E6, 0,  
0, NOTE_A6, 0, NOTE_B6,  
0, NOTE_AS6, NOTE_A6, 0,
```

```
NOTE_G6, NOTE_E7, NOTE_G7,  
NOTE_A7, 0, NOTE_F7, NOTE_G7,  
0, NOTE_E7, 0, NOTE_C7,  
NOTE_D7, NOTE_B6, 0, 0,
```

```
NOTE_C7, 0, 0, NOTE_G6,  
0, 0, NOTE_E6, 0,  
0, NOTE_A6, 0, NOTE_B6,  
0, NOTE_AS6, NOTE_A6, 0,
```

```
NOTE_G6, NOTE_E7, NOTE_G7,  
NOTE_A7, 0, NOTE_F7, NOTE_G7,  
0, NOTE_E7, 0, NOTE_C7,  
NOTE_D7, NOTE_B6, 0, 0
```

```
};
```

```
//Mario main them tempo
```

```
int tempo[] = {  
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,
```

```
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,
```

```
9, 9, 9,  
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,
```

```
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,
```

```
9, 9, 9,
```

```
12, 12, 12, 12,  
12, 12, 12, 12,  
12, 12, 12, 12,  
};
```

```
void setup() {  
  Serial.begin(9600);  
  irrecv.enableIRIn();  
  pinMode(4, OUTPUT);  
  pinMode(13, OUTPUT);  
  servoLeft.attach(servoLeftPin);  
  servoRight.attach(servoRightPin);  
}
```

```
void loop() {  
  if (irrecv.decode(&result)) {  
    irrecv.resume();  
  }  
  switch (result.value) {  
    case 0xFF629D: // Forward  
      Serial.println("Forward");  
      Serial.println();  
      digitalWrite(13, HIGH);  
      digitalWrite(4, HIGH);  
      delay(500);  
      digitalWrite(4, LOW);  
      digitalWrite(13, LOW);  
      servoLeft.writeMicroseconds(1685); // Left wheel counterclockwise  
      servoRight.writeMicroseconds(1300); // Right wheel clockwise  
      break;  
    case 0xFFA857: // Backwards  
      Serial.println("Backwards");  
      Serial.println();  
      digitalWrite(4, HIGH);  
      digitalWrite(13, HIGH);  
      delay(500);  
      digitalWrite(4, LOW);  
      digitalWrite(13, LOW);  
      servoLeft.writeMicroseconds(1300); // Left wheel clockwise  
      servoRight.writeMicroseconds(1685); // Right wheel counterclockwise  
      break;  
    case 0xFFC23D: // Right  
      Serial.println("Right");  
      Serial.println();  
      digitalWrite(4, HIGH);  
      digitalWrite(13, HIGH);  
      delay(500);  
      digitalWrite(4, LOW);
```

```

digitalWrite(13,LOW);
servoLeft.writeMicroseconds(1700); // Left wheel clockwise
servoRight.writeMicroseconds(1700); // Right wheel clockwise
delay(950);
break;
case 0xFF22DD: // Left
  Serial.println("Left");
  Serial.println();
  digitalWrite(4, HIGH);
  digitalWrite(13,HIGH);
  delay(500);
  digitalWrite(4, LOW);
  digitalWrite(13,LOW);
  servoLeft.writeMicroseconds(1300); // Left wheel counterclockwise
  servoRight.writeMicroseconds(1300); // Right wheel counterclockwise
  delay(950);
  break;
case 0xFF02FD: // OK (Stop)
  Serial.println("Stop");
  Serial.println();
  digitalWrite(4, HIGH);
  digitalWrite(13,HIGH);
  delay(500);
  digitalWrite(4, LOW);
  digitalWrite(13,LOW);
  servoLeft.writeMicroseconds(1500); // Left wheel stop
  servoRight.writeMicroseconds(1500); // Right wheel stop
  break;
case 0xFF6897: //remote 1
  Serial.println(" 'Mario Theme'");
  int size = sizeof(melody) / sizeof(int);
  for (int thisNote = 0; thisNote < size; thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / tempo[thisNote];

    buzz(melodyPin, melody[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);

    // stop the tone playing:
    buzz(melodyPin, 0, noteDuration);
  }
}

```

```
}  
}
```

```
void buzz(int targetPin, long frequency, long length) {  
  digitalWrite(13, HIGH);  
  long delayValue = 1000000 / frequency / 2; // calculate the delay value between transitions  
  /// 1 second's worth of microseconds, divided by the frequency, then split in half since  
  /// there are two phases to each cycle  
  long numCycles = frequency * length / 1000; // calculate the number of cycles for proper timing  
  /// multiply frequency, which is really cycles per second, by the number of seconds to  
  /// get the total number of cycles to produce  
  for (long i = 0; i < numCycles; i++) { // for the calculated length of time...  
    digitalWrite(targetPin, HIGH); // write the buzzer pin high to push out the diaphragm  
    delayMicroseconds(delayValue); // wait for the calculated delay value  
    digitalWrite(targetPin, LOW); // write the buzzer pin low to pull back the diaphragm  
    delayMicroseconds(delayValue); // wait again or the calculated delay value  
  }  
  digitalWrite(13, LOW);  
  
}
```

For the lab please create a program and wire the robot so that you can control it with the remote control to go Forward, Backwards, Left and Right plus it starts to navigate autonomously (use either IR or Sonar) when you press the button with the number one on the remote. Submit a report with your code and demo or sample video.

I am giving you one tip to keep in mind – using a while loop in the case

```
case 0xFF6897: //pressing number 1 on remote  
  while (true)  
    autonomous();  
  break;
```

End of Experiment