

8. Touch

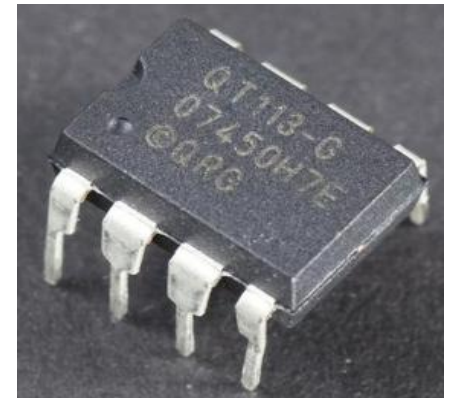


Touch

- **Touch is the most common way to tell a device what to do.**
 - ▣ Flip a switch to turn on the lights, press a button on a remote control, or click a key on your keyboard.
- **In this chapter:**
 - ▣ you'll meet a button, a squeeze sensor, and a capacitive touch sensor.
 - ▣ You'll learn to use pull-up resistors to avoid floating pins.
 - ▣ Finally, you'll build a haunted bell to try your touch skills in a project.

Touch

- A button is the most basic sensor, so it's also the most basic touch sensor. When a button is pressed, its leads are connected, closing a circuit. A microswitch is a type of small, durable button.
- A FlexiForce sensor detects how much force is being put on it. You can use it to test the might of your fingers or to detect when someone is sitting on a chair.
- A touch sensor senses touch without any moving parts. And the weird part of a touch sensor is that it doesn't even require touch! You'll learn to detect a hand placed on a table, through the table.



Contents

| | |
|---|--|
| 1 | Experiment: Button |
| 2 | Experiment: Microswitch |
| 3 | Potentiometer (Variable Resistor, Pot) |
| 4 | Sense Touch Without Touch (Capacitive Touch Sensor QT113) |
| 5 | Experiment: Feel the Pressure (FlexiForce) |
| 6 | Experiment: Build Your Own Touch Sensor |
| 7 | Test Project: Haunted Ringing Bell |

Experiment: Button

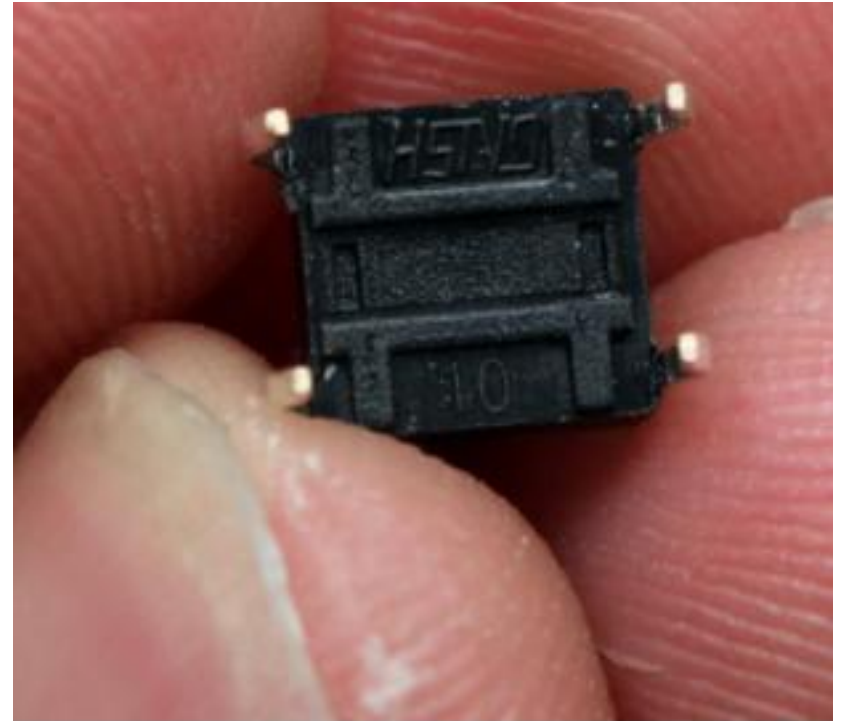
- A button (Figure 5-1) is the simplest sensor.
- Pressing the button down connects its leads, so that the button acts as a wire.
- Releasing the button breaks the circuit.
- All digital switch sensors (such as a reed switch or tilt switch) work like a button.



Figure 5-1. Push button

Experiment: Button

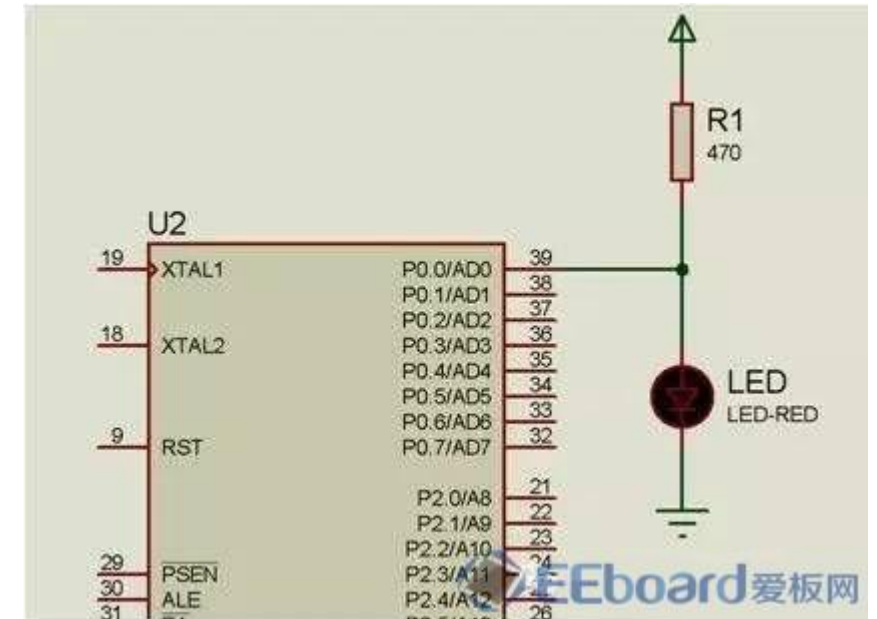
- Buttons come in many sizes and forms.
- When using a breadboard, it's convenient to use a button with four leads.
- The leads are in pairs, so that two adjacent leads are always connected to each other.
- When you turn the button upside-down, you can see which leads are connected. There is a beveled line showing the connected leads (Figure 5-2).



5-2. Bottom of a push button

Pull-Up Resistor

- To get a reading from a data pin, it must be connected somewhere. You should not read an unconnected floating pin. A pull-up resistor pulls a pin HIGH when it's not connected to anything else, avoiding the floating state.
- The state of a data pin can be read with commands such as `digitalRead()`, `analogRead()`, `botbook_gpio.read()`, `botbook_mcp2003.readAnalog()`.



Pull-Up Resistor

- A pin that's connected to ground or HIGH is an obvious case: when you read its state, it's clear which state it's in. If the pin is connected to ground (GND, 0 V), a digital read returns LOW. If the pin is connected to HIGH (+5 V or +3.3 V), the read returns HIGH.
- If you read an unconnected, floating data pin, you would get an undefined value. It could be HIGH, it could be LOW, it could change every second or stay the same forever. Such an undefined value is completely useless, so there is no point in reading a floating pin.

Pull-Up Resistor

- Consider a circuit with a button between a data pin and ground (Figure 5-3).
When the button is pressed, the data pin is connected to ground so it's LOW.
- What about when the button is up? You must use a pull-up resistor to bring the data pin HIGH.

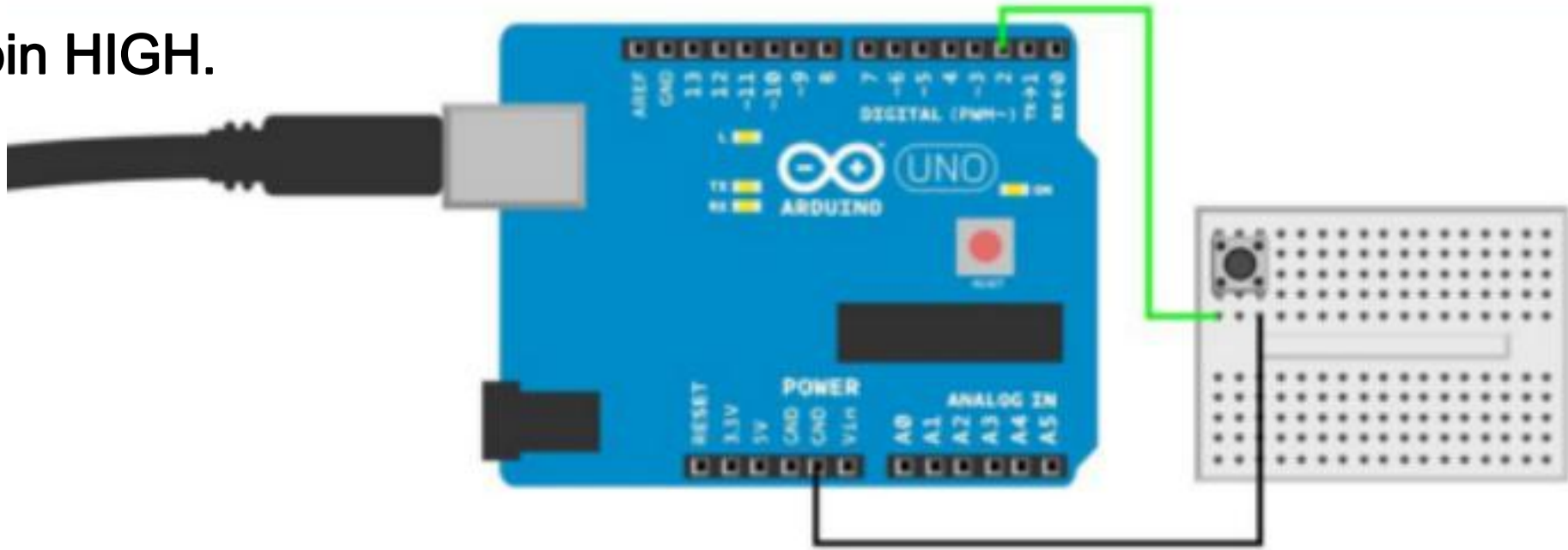
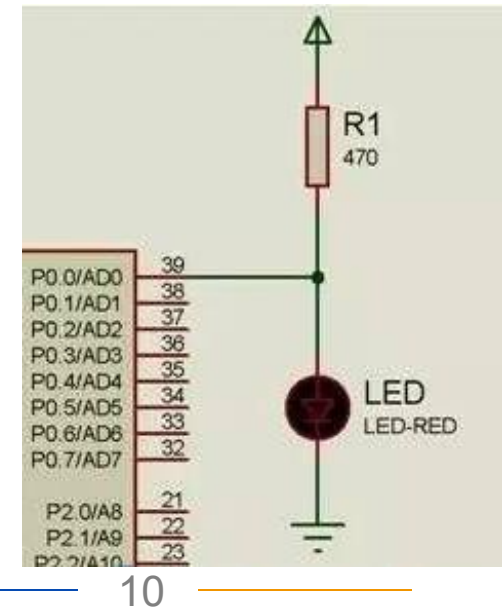


Figure 5-3. Button circuit for Arduino

Pull-Up Resistor

- The pull-up resistor is big. We often use one in the tens of thousands ohms range. This way, when the button is pressed and the data pin is connected to both ground and the pull-up, there is no short-circuit between +5 V and ground because the path of least resistance is between the data pin and ground, rather than between +5 V and ground.
- For convenience, the code below uses built-in pull-up resistors.
 - ❑ For Arduino, you can enable the onboard pull-up resistors with a line of code.
 - ❑ For Raspberry Pi, you'll use one of the pins that has an always-on pull-up resistor.



Code and Connection for Arduino

Example 5-1. button.ino

// button.ino - light an LED by pressing a button

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
int buttonPin=2;  
int ledPin=13;  
int buttonStatus=-1;
```

```
void setup() {  
    pinMode(ledPin, OUTPUT);  
    pinMode(buttonPin, INPUT); // ❶  
    digitalWrite(buttonPin, HIGH); // ❷  
}
```

`pinMode(buttonPin, INPUT_PULLUP)`



```
void loop() {  
    buttonStatus=digitalRead(buttonPin); //❸  
    if (LOW==buttonStatus) { //❹  
        digitalWrite(ledPin, HIGH); //❺  
    } else {  
        digitalWrite(ledPin, LOW);  
    }  
    delay(20); //❻  
}
```

Code and Connection for Arduino

Example 5-1. button.ino

// button.ino - light an LED by pressing a button

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
int buttonPin=2;  
int ledPin=13;  
int buttonStatus=-1;
```

```
void setup() {  
    pinMode(ledPin, OUTPUT);  
    pinMode(buttonPin, INPUT);    // ❶  
    digitalWrite(buttonPin, HIGH); // ❷  
}
```



```
void loop() {  
    buttonStatus=digitalRead(buttonPin);    //❸  
    if (LOW==buttonStatus) {                //❹  
        digitalWrite(ledPin, HIGH);        //❺  
    } else  
    {  
        digitalWrite(ledPin, LOW);  
    }  
    delay(20);                               //❻  
}
```

Code and Connection for Arduino

- This code keeps the LED lit when the button is held down.
- If you want to instead *toggle* the LED each time you press the button, you must *debounce* the input.
- For example, creating a device where one click turns the LED on and the next turns it off requires *debouncing*.

Code and Connection for Raspberry Pi

- Wire up the Raspberry Pi as shown in shown in Example 5-2.

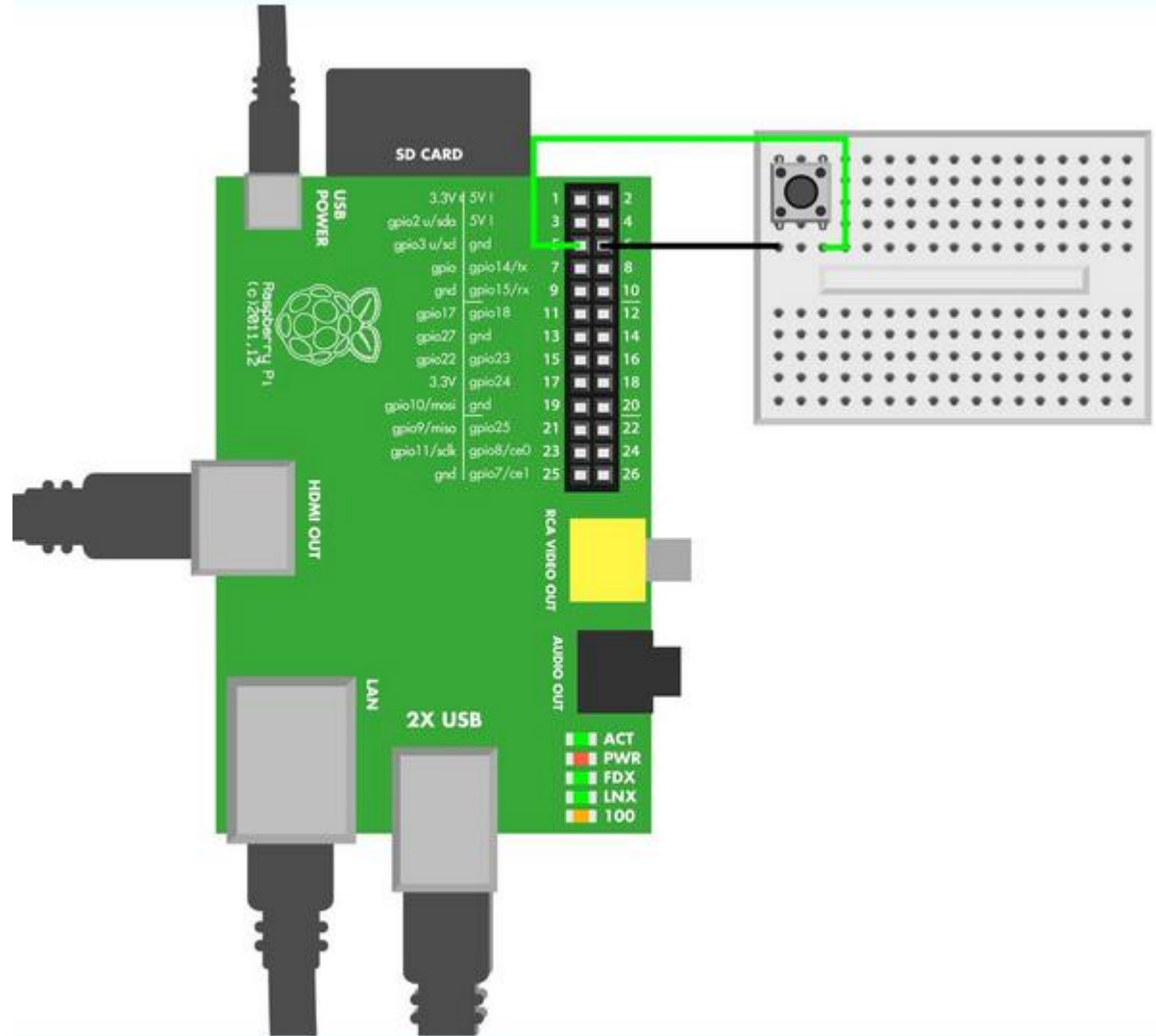


Figure 5-4. Button circuit for Raspberry Pi

Code and Connection for Raspberry Pi

Example 5-2. button.py

button.py - write to screen if button is down or up

(c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import time
import botbook_gpio as gpio          # ❶

def main():
    buttonpin = 3  # has internal pull-up  # ❷
    gpio.mode(buttonpin, "in")           # ❸

    while (True):                        # ❹
        buttonUp = gpio.read(buttonpin)  # ❺
        if(buttonUp == gpio.HIGH):
            print "Button is up"
        else:
            print "Button is pressed"
            time.sleep(0.3) # seconds    # ❻

if __name__ == "__main__":
    main()
```

Contents

| | |
|---|--|
| 1 | Experiment: Button |
| 2 | Experiment: Microswitch |
| 3 | Potentiometer (Variable Resistor, Pot) |
| 4 | Sense Touch Without Touch (Capacitive Touch Sensor QT113) |
| 5 | Experiment: Feel the Pressure (FlexiForce) |
| 6 | Experiment: Build Your Own Touch Sensor |
| 7 | Test Project: Haunted Ringing Bell |

Experiment: Microswitch

- A microswitch is a button (see [Figure 5-5](#)). When you press the button, the leads are connected. This experiment writes “0” to the serial port when the microswitch is pressed.
- Microswitches are popular because of their low price, small size, and durability.
- A typical microswitch will last more than a million presses.
- The clicky sound and the feel of the click come from a tiny arm turning around a pivot.

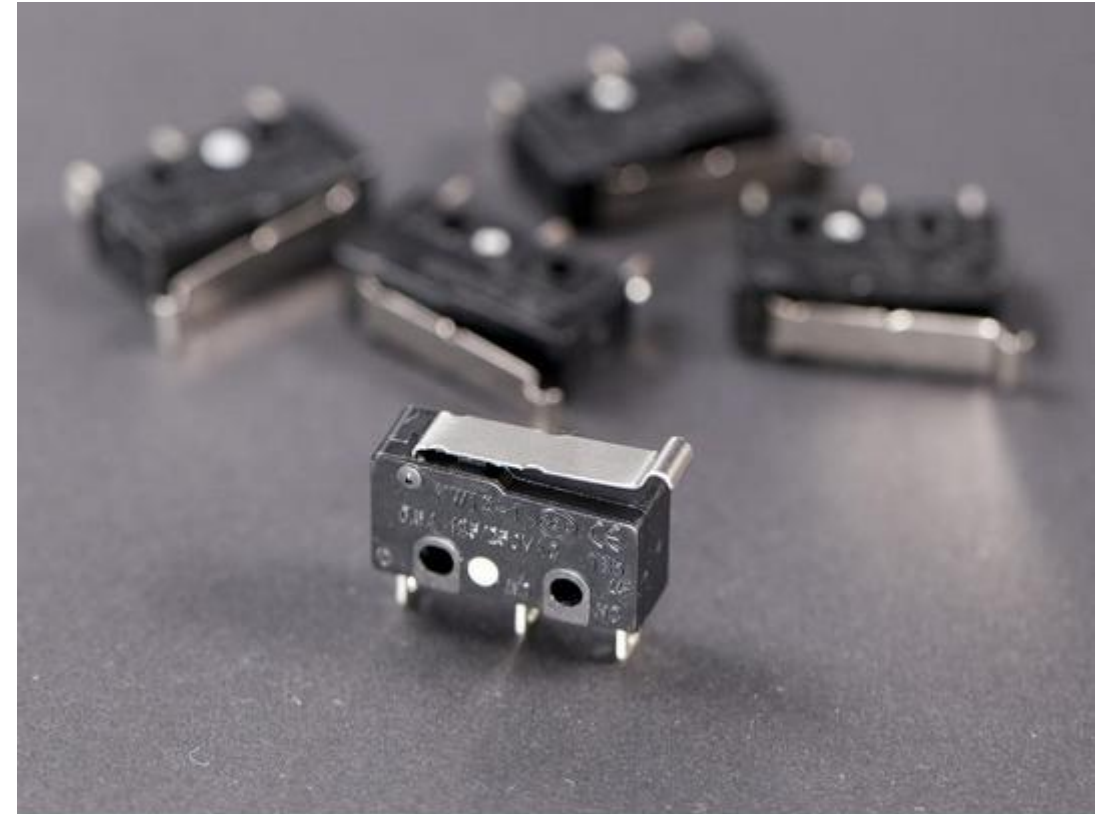
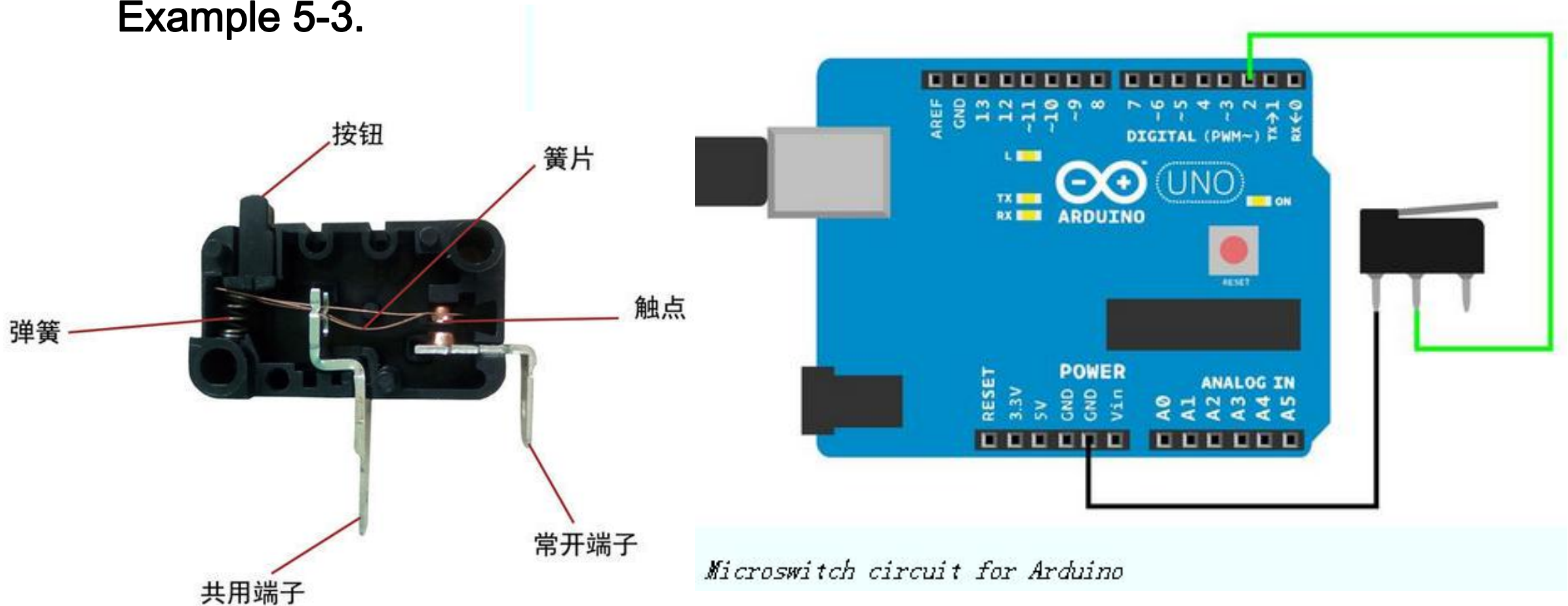


Figure 5-5. Microswitch

Microswitch Code and Connection for Arduino

- Build the circuit shown in Figure 5-6 and upload the code shown in Example 5-3.



Microswitch Code and Connection for Arduino

Example 5-3. microswitch.ino

// microswitch.ino - print to serial if microswitch is down or up

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
const int switchPin = 2;
int switchState = -1;           //❶
void setup() {
    Serial.begin(115200);
    pinMode(switchPin, INPUT);
    digitalWrite(switchPin, HIGH); //❷ internal pull-up
}
void loop() {
    switchState = digitalRead(switchPin);
    Serial.println(switchState);   //❸
    delay(10);
}
```

Experiment: Microswitch

- You can make simple feeler antennas for your robot from a microswitch just by hot gluing a zip tie to it (Figure 5-7).



Figure 5-7. Feeler antenna from microswitch

Microswitch Code and Connection for Raspberry Pi

- Wire up the Raspberry Pi as shown in [Figure 5-8](#) shown in [Example 5-4](#).

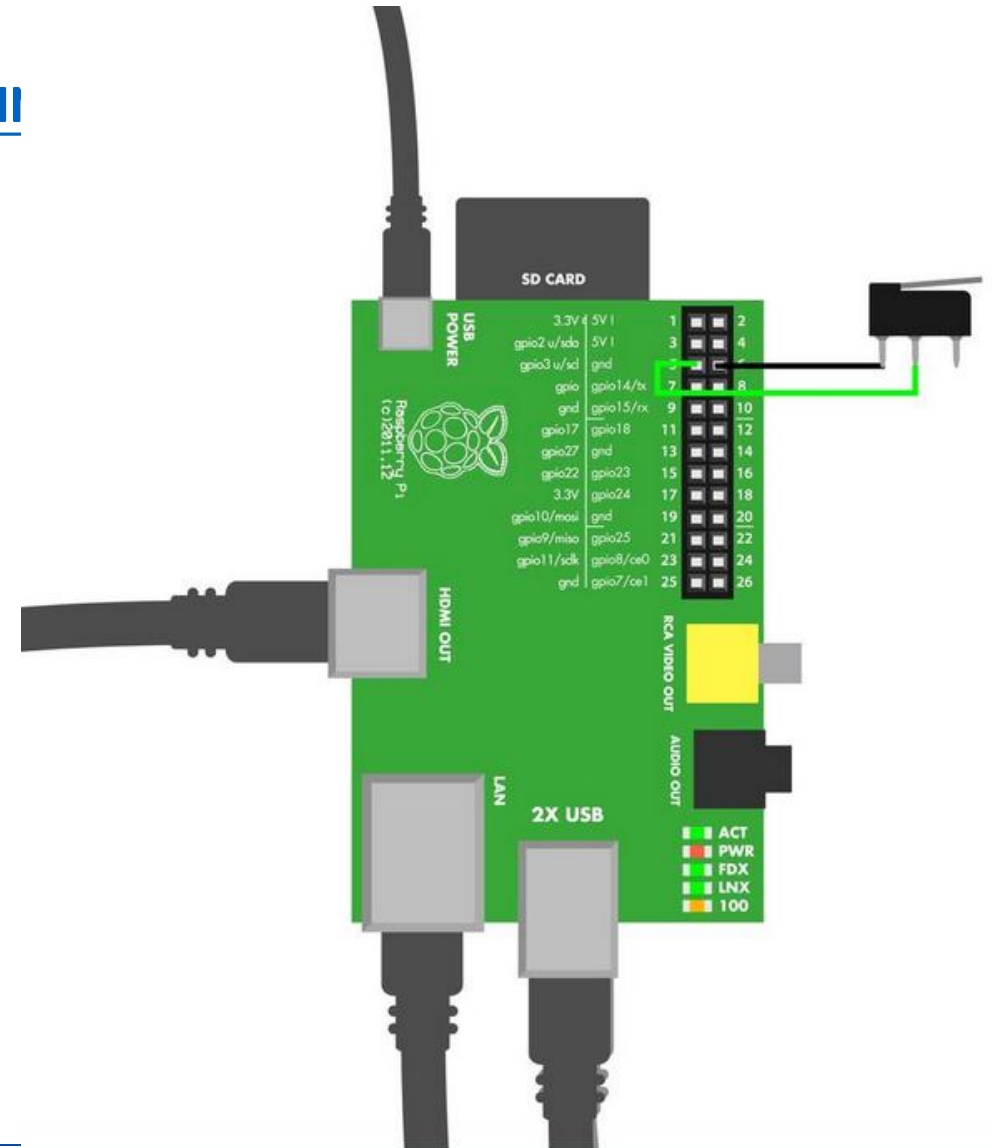


Figure 5-8. Microswitch circuit for Raspberry Pi

Microswitch Code and Connection for Raspberry Pi

Example 5-4. microswitch.py

microswitch.py - write to screen if switch is pressed or not

(c) BotBook.com - Karvinen, Karvinen, Valtokari

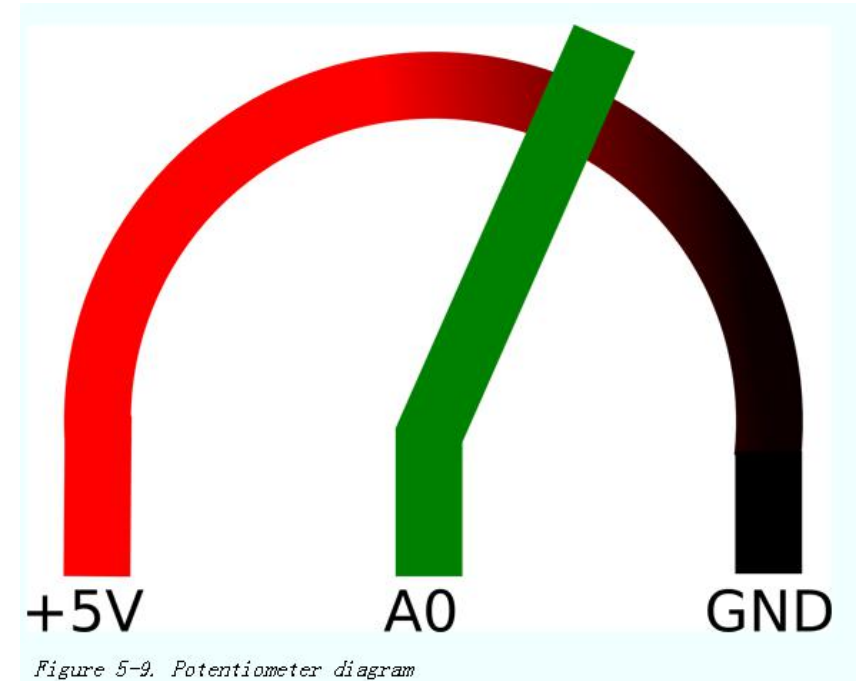
```
import time
import botbook_gpio as gpio          # ❶
def main():
    switchpin = 3  # has internal pull-up    # ❷
    gpio.mode(switchpin, "in")
    while (True):
        switchState = gpio.read(switchpin)    # ❸
        if(switchState == gpio.LOW):
            print "Switch is pressed"
        else:
            print "Switch is up"
            time.sleep(0.3) # seconds
if __name__ == "main__":
    main()
```

Contents

| | |
|---|--|
| 1 | Experiment: Button |
| 2 | Experiment: Microswitch |
| 3 | Potentiometer (Variable Resistor, Pot) |
| 4 | Sense Touch Without Touch (Capacitive Touch Sensor QT113) |
| 5 | Experiment: Feel the Pressure (FlexiForce) |
| 6 | Experiment: Build Your Own Touch Sensor |
| 7 | Test Project: Haunted Ringing Bell |

Experiment: Potentiometer (Variable Resistor, Pot)

- In this experiment, you'll control how quickly an LED blinks by turning a potentiometer.
- A potentiometer is a variable resistor. You turn its knob to change its resistance.
- A normal, non-variable resistor has just two leads. If you want to use a potentiometer in place of a single resistor, use the middle lead and either of the side leads.



Potentiometer Code and Connection for Arduino

- Figure 5-10 shows the connection diagram for Arduino. Set up your circuit as shown, and then upload and run the sketch shown in Example 5-5.

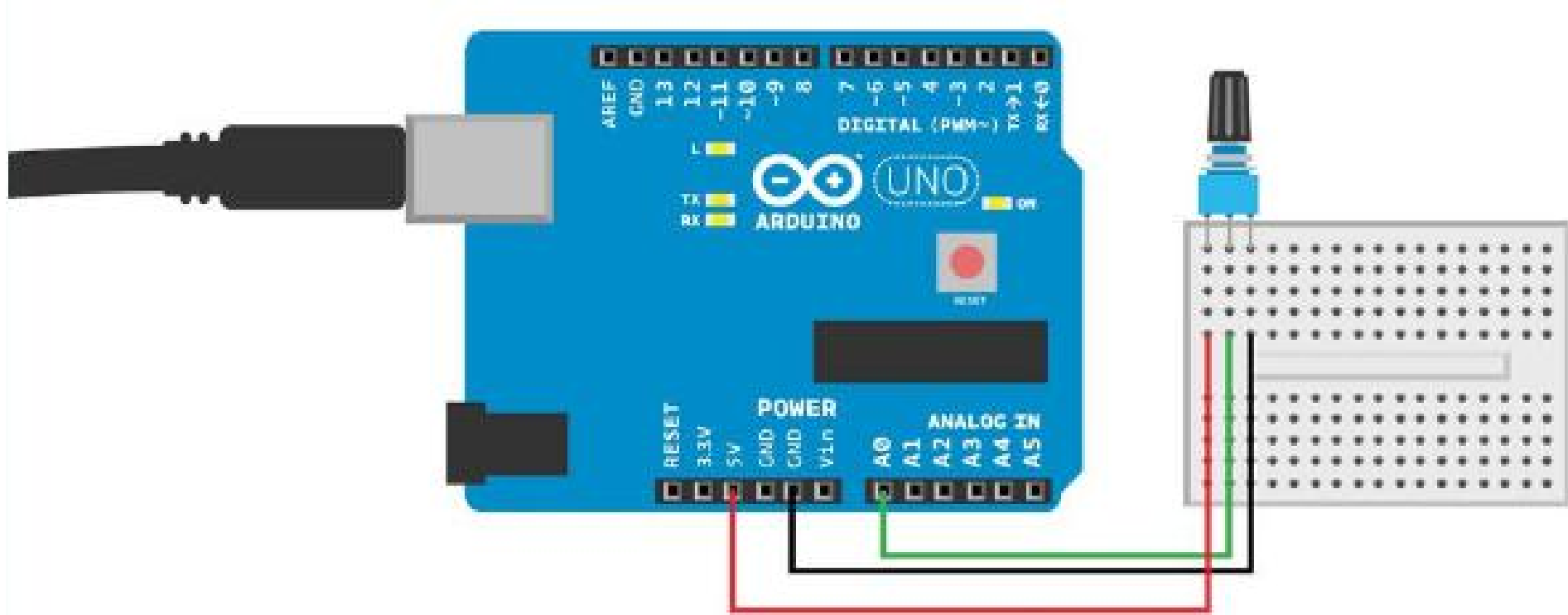


Figure 5-10. Potentiometer circuit for Arduino

Potentiometer Code and Connection **for Arduino**

Example 5-5. pot.ino

*// pot.ino - control LED blinking speed with potentiometer
// (c) BotBook.com - Karvinen, Karvinen, Valtokari*

```
int potPin=A0;    // ❶
int ledPin=13;
int x=0;          // 0..1023
void setup() {
    pinMode(ledPin, OUTPUT);
    pinMode(potPin, INPUT);
}
void loop() {
    x=analogRead(potPin);        // ❷
    digitalWrite(ledPin, HIGH);
    delay(x/10);                 // ms    // ❸
    digitalWrite(ledPin, LOW);
    delay(x/10);
}
```

Potentiometer Code and Connection for Raspberry Pi

- Figure 5-11 shows the circuit diagram for using a pot with Raspberry Pi.
- Wire it up as shown, and run the program shown in Example 5-6.

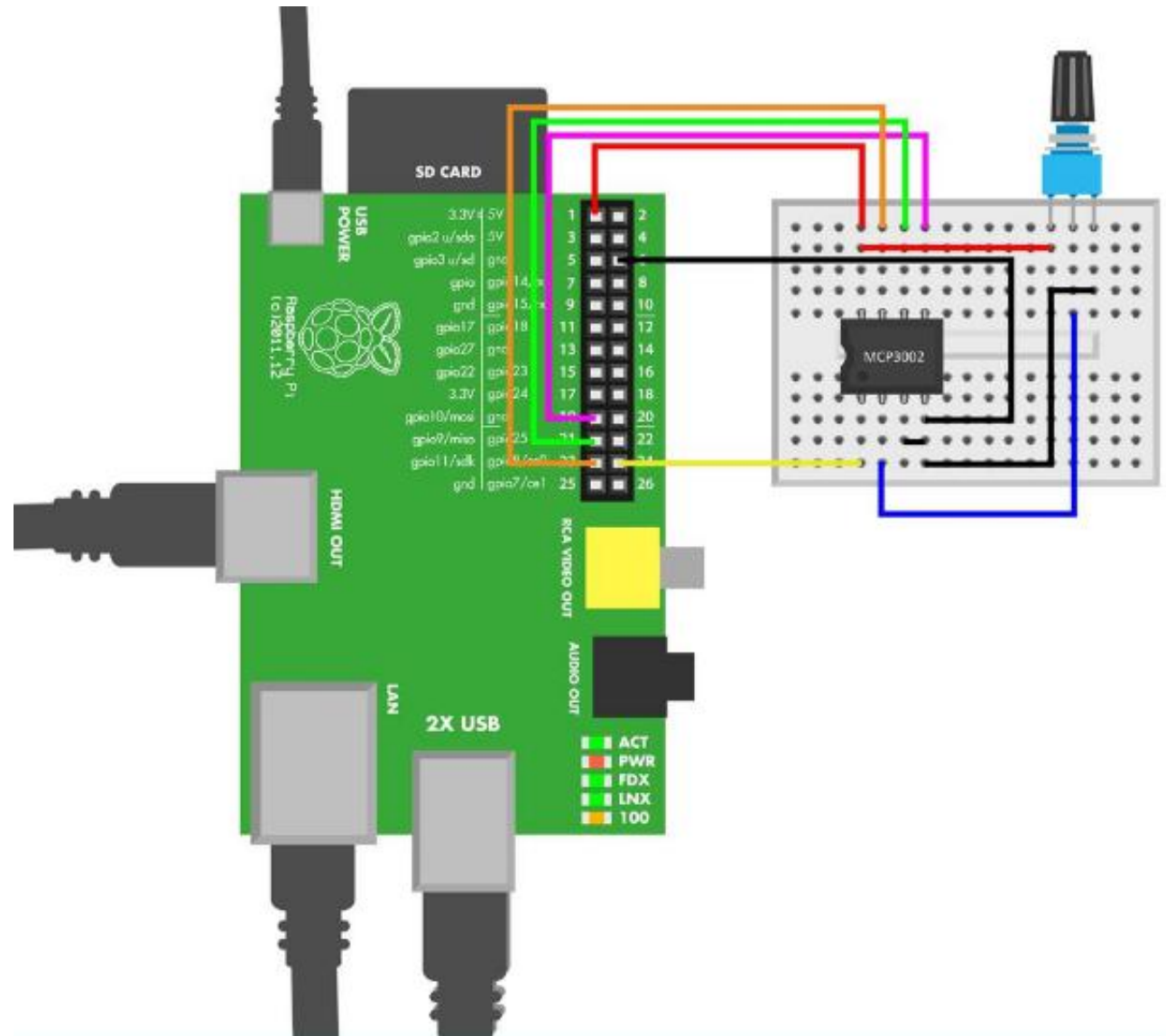


Figure 5-11. Potentiometer circuit for Raspberry Pi

Potentiometer Code and Connection **for Raspberry Pi**

Example 5-6. pot.py

pot.py - potentiometer with mcp3002

(c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import spidev                # ❶
import time
def readPotentiometer():
    spi = spidev.SpiDev()     # ❷
    spi.open(0, 0)            # ❸
    command = [1, 128, 0]     # ❹
    reply = spi.xfer2(command) # ❺
    #Parse reply 10 bits from 24 bit package
    data = reply[1] & 31       # ❻
    data = data << 6           # ❼
    data = data + (reply[2] >> 2) # ❽
    spi.close()               # ❾
    return data
```

```
def main():
    while True:
        potentiometer = readPotentiometer() # ❿
        print("Current potentiometer value is %i "
              % potentiometer)
        time.sleep(0.5)

if __name__ == "__main__":
    main()
```

Contents

| | |
|---|--|
| 1 | Experiment: Button |
| 2 | Experiment: Microswitch |
| 3 | Potentiometer (Variable Resistor, Pot) |
| 4 | Sense Touch Without Touch (Capacitive Touch Sensor QT113) |
| 5 | Experiment: Feel the Pressure (FlexiForce) |
| 6 | Experiment: Build Your Own Touch Sensor |
| 7 | Test Project: Haunted Ringing Bell |

Experiment: Sense Touch Without Touch (Capacitive Touch Sensor QT113)

- The secret of a capacitive touch sensor is that it doesn't sense touch at all.
- Instead, it measures how long it takes to load a piece of wire electrically.
- If there is a human (a big sack of water) nearby, it takes longer to load the wire.
- The QT113 is an IC (integrated chip) for capacitive sensing. The protocol is simple: when a touch is detected, the output pin goes LOW.

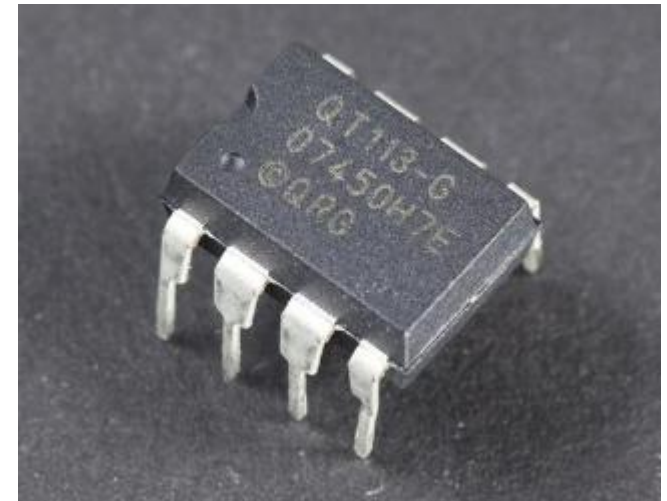
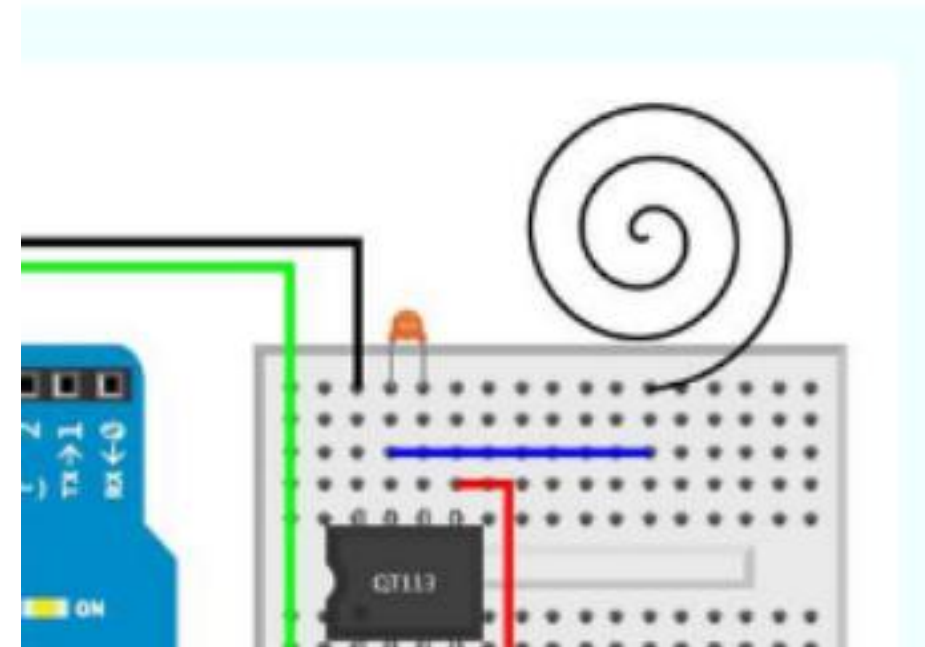


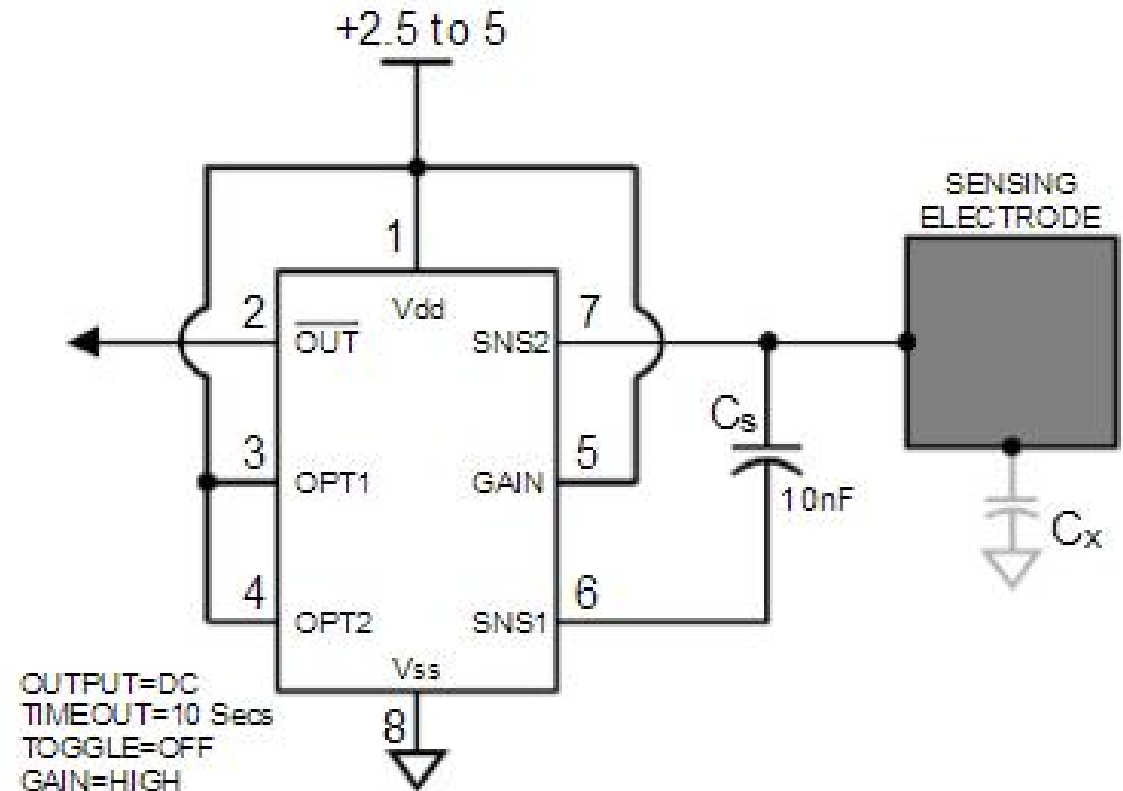
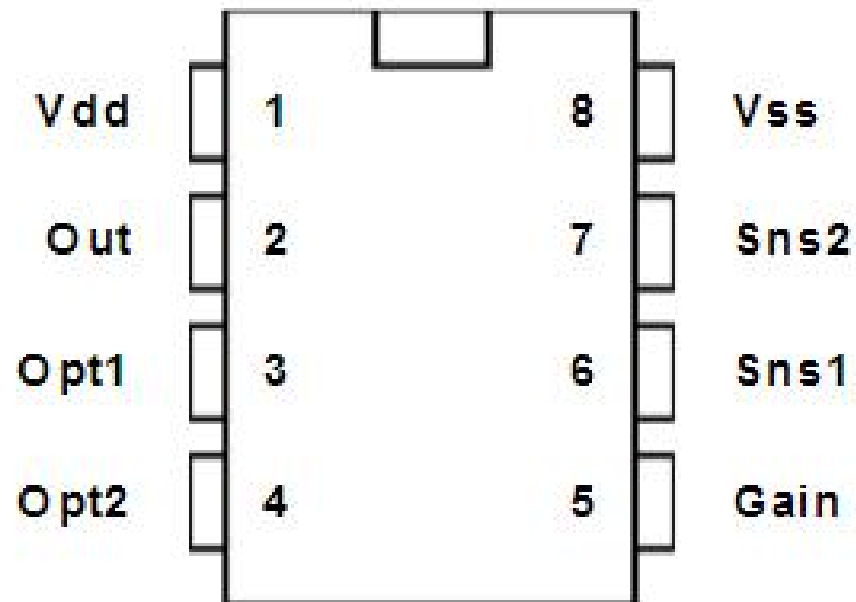
Figure 5-12. QT113 capacitive touch sensor IC

Experiment: Sense Touch Without Touch (Capacitive Touch Sensor QT113)

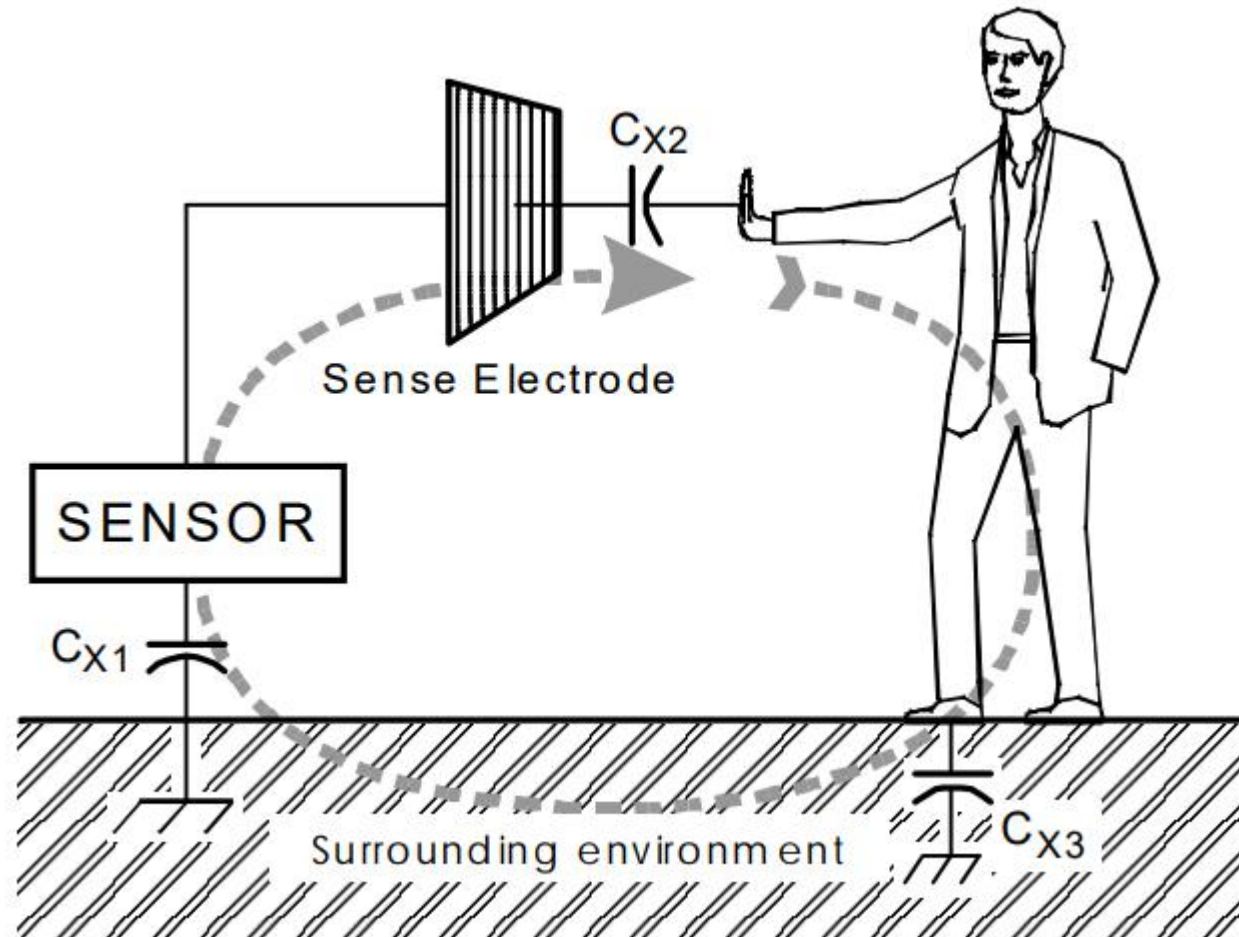
- A good capacitive sensor needs some kind of surface.
- In this experiment, you'll use a piece of metal wire turned into a spiral. A piece of aluminum foil could also work.
- The circuit also uses a 10-500 nF capacitor.
- There are many ways to do capacitive sensing:
 - A piece of wire and a simple timer
 - A piece of wire and CapSense library
 - A specialized chip (like QT113)



QT113 Code and Connection for Arduino



QT113 Code and Connection for Arduino



QT113 Code and Connection for Arduino

- Figure 5-13 shows the circuit for using the QT113 with Arduino. Hook it up as shown, and then run the sketch shown in Example 5-7.

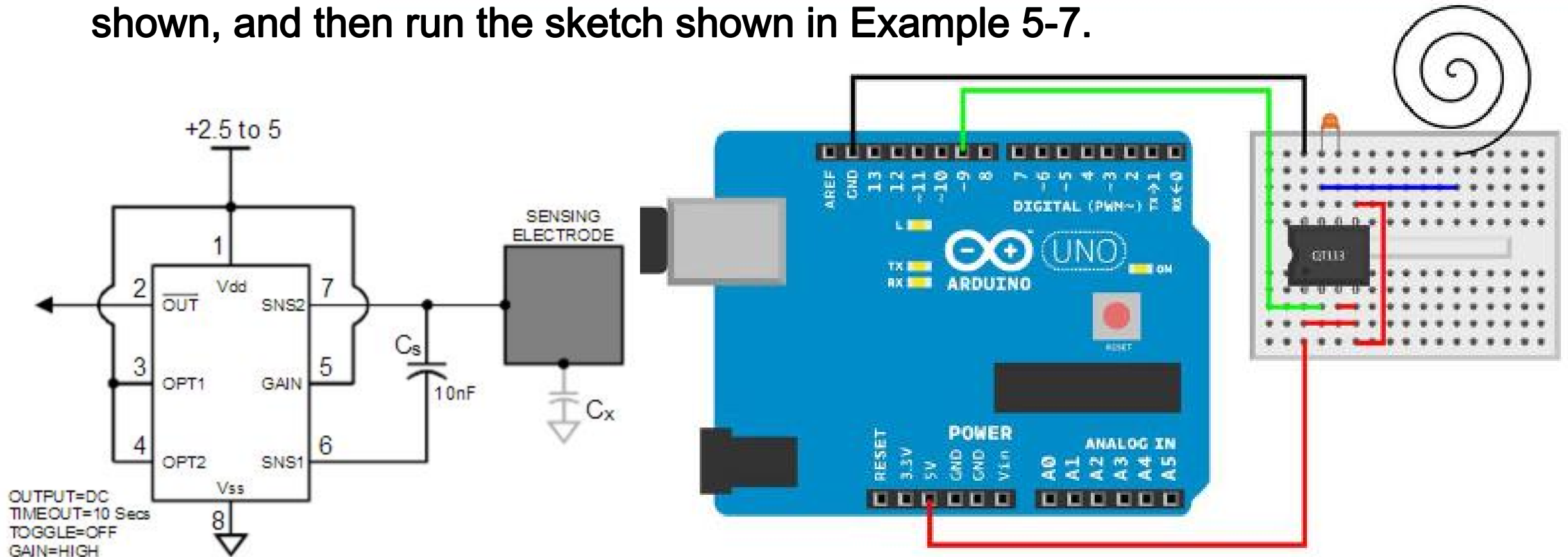


Figure 5-13. QT113 circuit for Arduino

QT113 Code and Connection for Arduino

Example 5-7. qt113.ino

// qt113.ino - qt113 touch sensor

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
int sensorPin = 9;
void setup(){
    pinMode(sensorPin, INPUT);
    Serial.begin(115200);
}

void loop(){
    int touch = digitalRead(sensorPin);
    if(touch == LOW) {
        Serial.println("Touch detected");
    } else {
        Serial.println("No Touch detected");
    }
    delay(100);
}
```

//这是一个简单的数字开关传感器

QT113 Code and Connection for Raspberry Pi

- Figure 5-14 shows the connections for using Raspberry Pi with the QT113.
- Wire it up and run the program shown in Example 5-8.

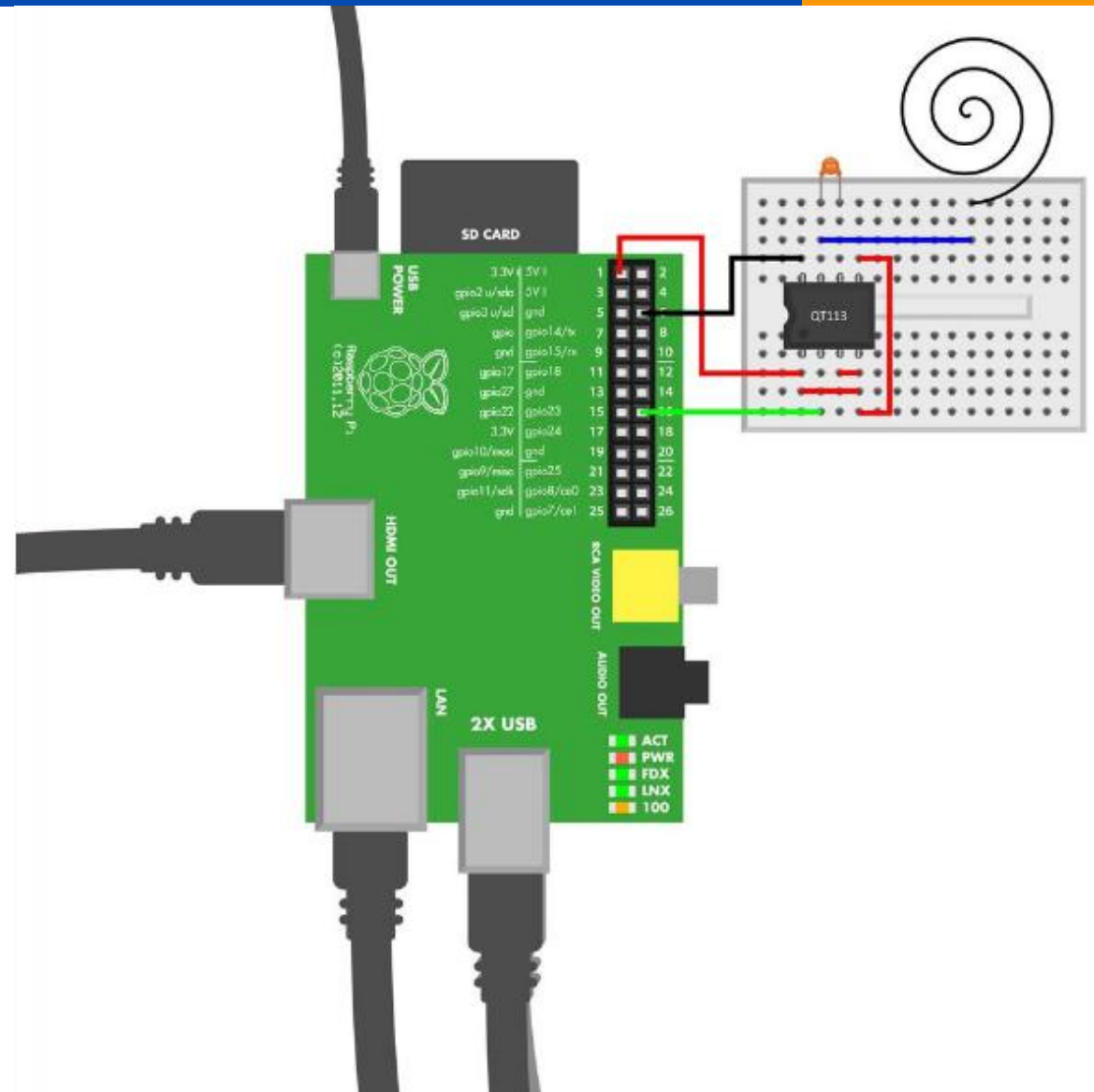


Figure 5-14. QT113 circuit for Raspberry Pi

QT113 Code and Connection for Raspberry Pi

Example 5-8. qt113.py

qt113.py - read touch information from QT113

(c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import time
import botbook_gpio as gpio
def main():
    limitPin = 23
    gpio.setMode(limitPin, "in")
    while True:
        if gpio.read(limitPin) == gpio.LOW:    #这是一个简单的数字开关传感器
            print("Touch detected!")
            time.sleep(0.5)

if __name__ == "__main__":
    main()
```

Environment Experiment: Sensing Touch Through Wood



Figure 5-15. For the user, the sensor looks like wood



Figure 5-16. Touch wire spiral, hidden from user

Contents

| | |
|---|--|
| 1 | Experiment: Button |
| 2 | Experiment: Microswitch |
| 3 | Potentiometer (Variable Resistor, Pot) |
| 4 | Sense Touch Without Touch (Capacitive Touch Sensor QT113) |
| 5 | Experiment: Feel the Pressure (FlexiForce) |
| 6 | Experiment: Build Your Own Touch Sensor |
| 7 | Test Project: Haunted Ringing Bell |

Experiment: Feel the Pressure (FlexiForce)

- The FlexiForce sensors measures squeeze (pressure) on its round head (Figure 5-17).



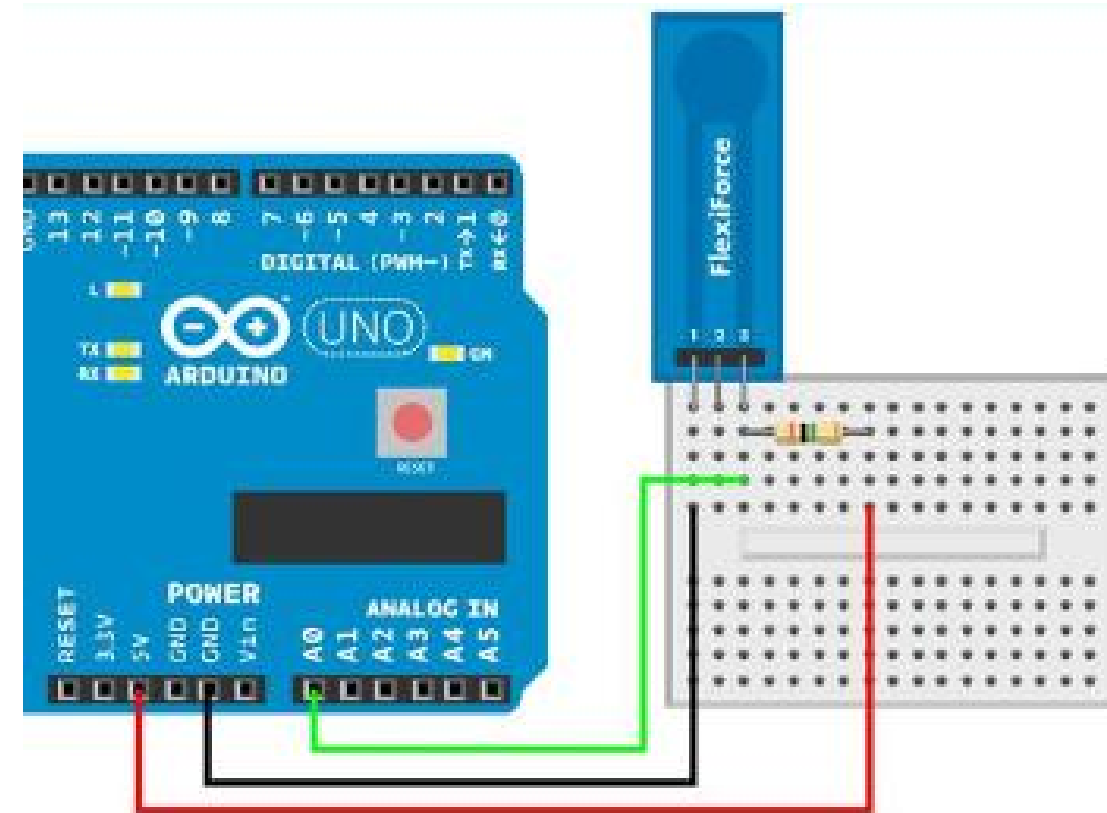
- FlexiForce is a simple analog resistance sensor. The more you squeeze, the lower the resistance.
- It has just two leads. The third lead in the middle is not connected, but is there just to make connecting it easier.
- As it's just a resistor, it has no polarity, and it doesn't matter which way you connect it. You can even test it with a multimeter that's on the resistance setting.

Experiment: Feel the Pressure (FlexiForce)

- You can use FlexiForce for
 - ▣ an alarm that silences only when you get out of bed,
 - ▣ a squeezing strong man competition,
 - ▣ and for measuring squeezing precision.
- If you need a larger area for stepping over, you can put a piece of wood over the FlexiForce.

FlexiForce Code and Connection for Arduino

- FlexiForce is an analog resistance sensor. As Arduino has a built-in analog-to-digital converter, reading the value is a simple call to `analogRead()`.
- A 1 mega ohm pull-up resistor is used to avoid a floating pin.
- Figure 5-18 shows the wiring diagram, and Example 5-9 shows the sketch. After you wire up the Arduino, load the sketch and run it.



in for Arduino

FlexiForce Code and Connection for Arduino

Example 5-9. flexiforce_25.ino

// flexiforce_25.ino - send flexiforce squeeze values to computer serial monitor

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
int squeezePin=A0;           // ❶
int x=-1;                    // 0..1023
void setup() {
    pinMode(squeezePin, INPUT);
    Serial.begin(115200); // bit/s
}

void loop() {
    x=analogRead(squeezePin); //❷
    Serial.println(x);
    delay(500); // ms
}
```

FlexiForce Code and Connection for Raspberry Pi

- The Raspberry Pi connection is similar to other analog resistance sensors (Figure 5-19).

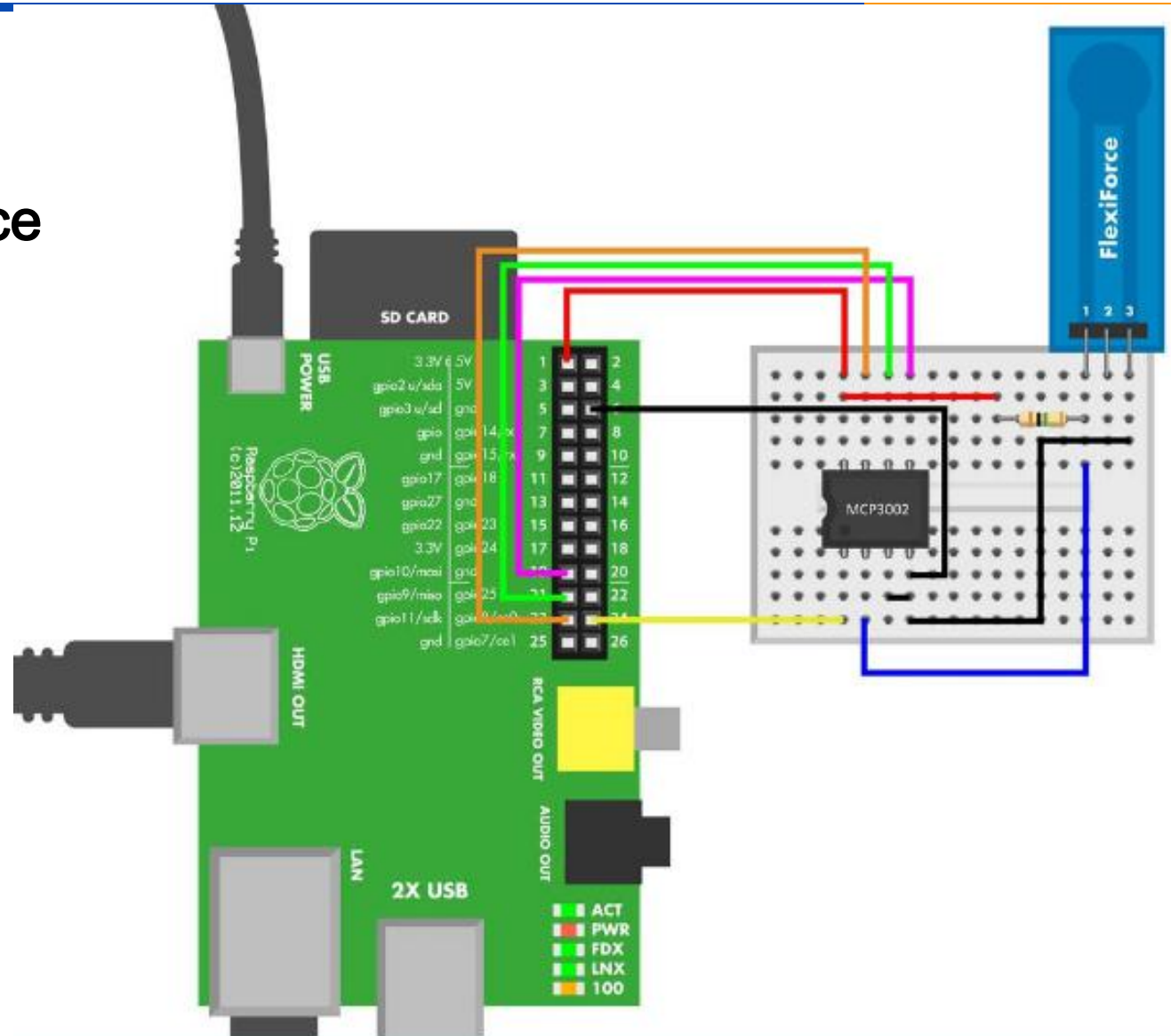
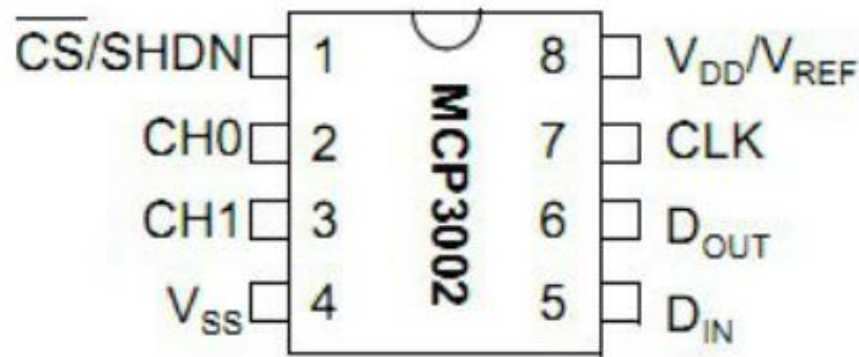


Figure 5-19. FlexiForce connection for Raspberry Pi

FlexiForce Code and Connection for Raspberry Pi

Example 5-10. flexiforce.py

flexiforce.py - sense force and print value to screen.

(c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import time
import botbook_mcp3002 as mcp          # ❶
def readFlexiForce():
    return mcp.readAnalog()            # ❷

def main():
    while True:                         # ❸
        f = readFlexiForce()           # ❹
        print("Current force is %i " % f) # ❺
        time.sleep(0.5)                 # s    # ❻

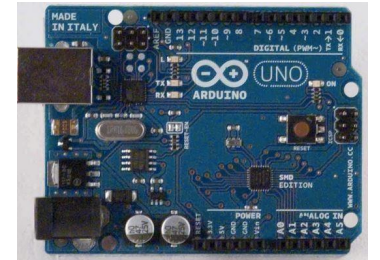
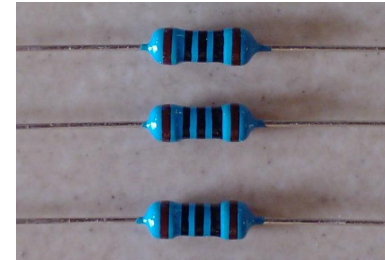
if __name__ == "__main__":
    main()
```

Contents

| | |
|---|--|
| 1 | Experiment: Button |
| 2 | Experiment: Microswitch |
| 3 | Potentiometer (Variable Resistor, Pot) |
| 4 | Sense Touch Without Touch (Capacitive Touch Sensor QT113) |
| 5 | Experiment: Feel the Pressure (FlexiForce) |
| 6 | Experiment: Build Your Own Touch Sensor |
| 7 | Test Project: Haunted Ringing Bell |

Experiment: Build Your Own Touch Sensor

- If capacitive sensing is just measuring the time it takes to load an electrical charge, could you build one yourself, and avoid using the QT113 chip?
- It's possible to sense touch with just aluminum foil, a resistor, and good ground for your Arduino board.



- Capacitive sensing is a practical matter. Even though the principle is simple, implementation is precise business. In addition to good grounding, the measurements must use sliding averages to smooth out any random fluctuation.

Experiment: Build Your Own Touch Sensor

- For reliable ground, the power source of Arduino must be connected to wall socket.
- The measuring part, aluminum foil, is connected between two data pins. Connecting something between data pins like this is quite rare—you usually connect one wire to ground or +5 V and the other to a data pin.

Experiment: Build Your Own Touch Sensor

- The smaller 10 kOhm resistor (brown-black-orange) helps protect against static electricity.
- The big 1 MOhm to 50 MOhm resistor selects the sensitivity. The bigger the resistor, the farther away it detects a human. The trade-off for bigger detection distance is a slower reading speed.

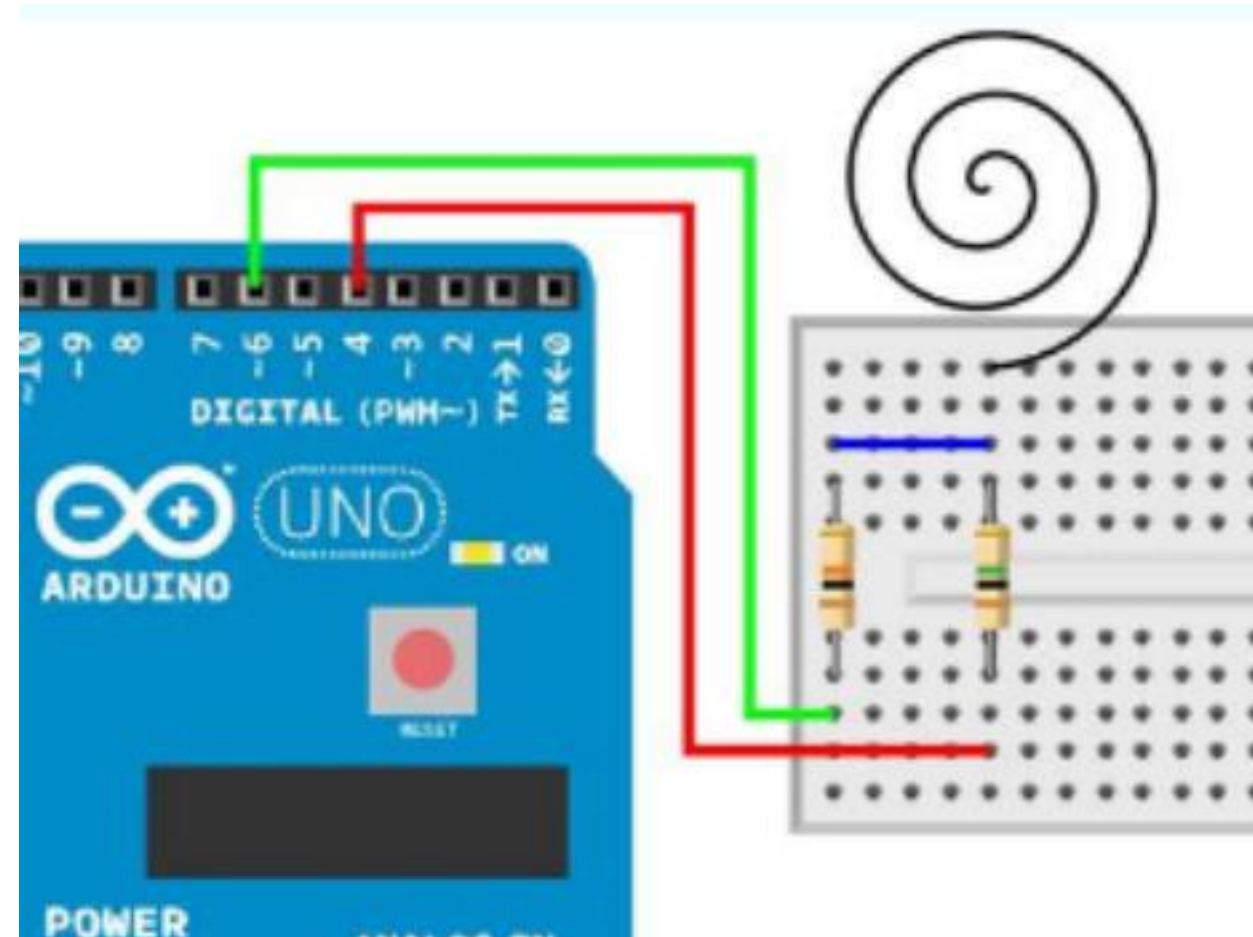


Figure 5-20. Capacity sensor connection for Arduino

Experiment: Build Your Own Touch Sensor

Example 5-11. diy_capacitive_sensor.ino

// diy_capacitive_sensor.ino - measure touch

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
const int sendPin = 4;
```

```
const int readPin = 6;
```

```
void setup() {
```

```
    Serial.begin(115200);
```

```
    pinMode(sendPin,OUTPUT);
```

```
    pinMode(readPin,INPUT);
```

```
    digitalWrite(readPin,LOW);
```

```
}
```

```
void loop() {
```

```
    int time = 0;
```

```
    digitalWrite(sendPin,HIGH);
```

```
    while(digitalRead(readPin) == LOW) time++;
```

```
    Serial.println(time);
```

```
    digitalWrite(sendPin,LOW);
```

```
    delay(100);
```

```
}
```



Capsense Code and Connection for Raspberry Pi

- Figure 5-21 shows the wiring diagram for Raspberry Pi. Wire it up as shown, and then run the code shown in Example 5-12.

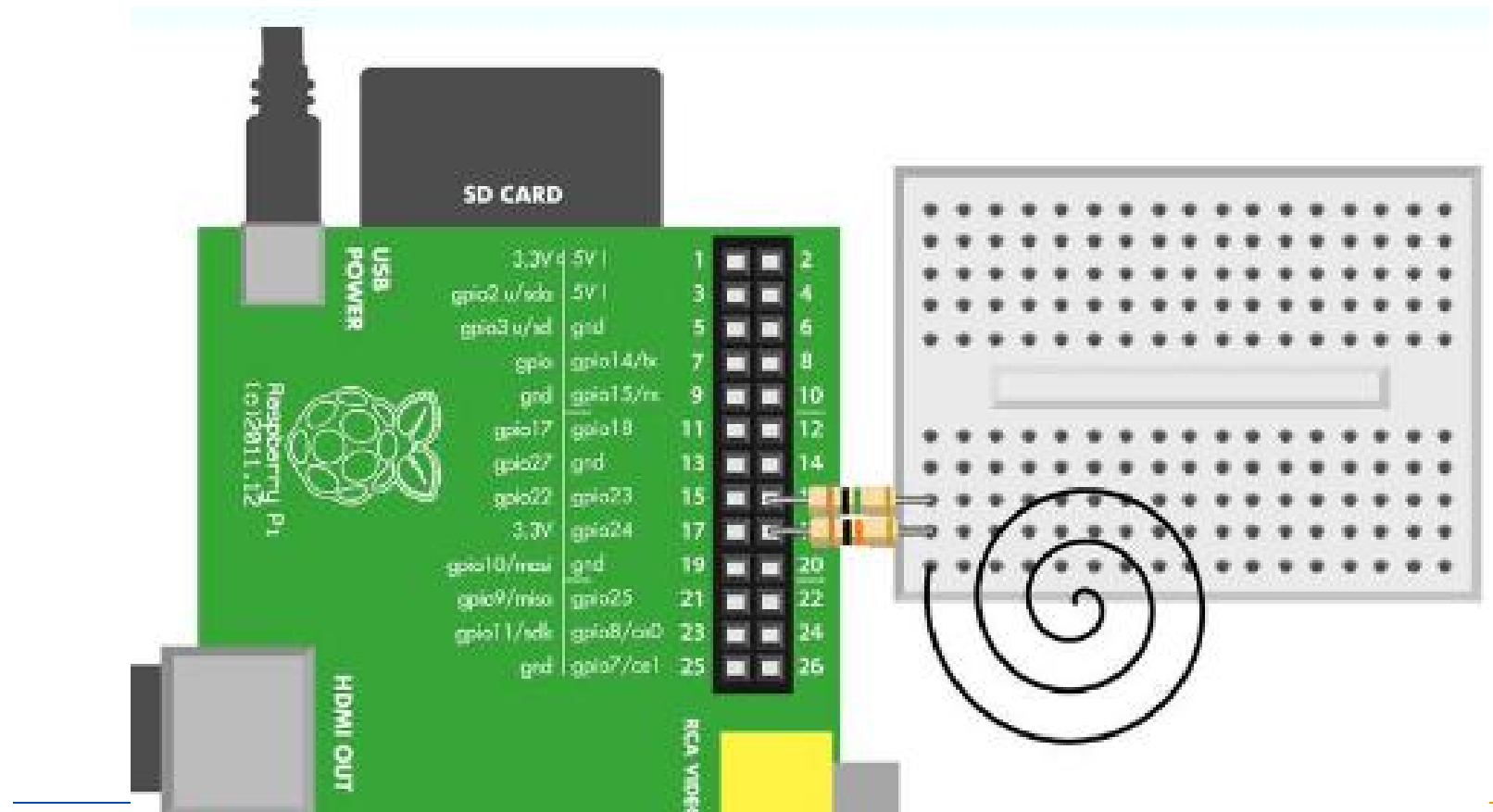


Figure 5-21. Capacity sensor connection for Raspberry Pi

Capsense Code and Connection for Raspberry Pi

Example 5-12. diy_capacity_sensor_simple.py

diy_capacity_sensor_simple.py - read touch from diy capacity sensor.

(c) BotBook.com - Karvinen, Karvinen, Valtokari

```
import time
import botbook_gpio as gpio
```

```
def main():
    while True:
        touch = sample(30)
        print("Touch: %d" % touch)
        time.sleep(0.5)
```

```
if __name__ == "__main__":
    main()
```

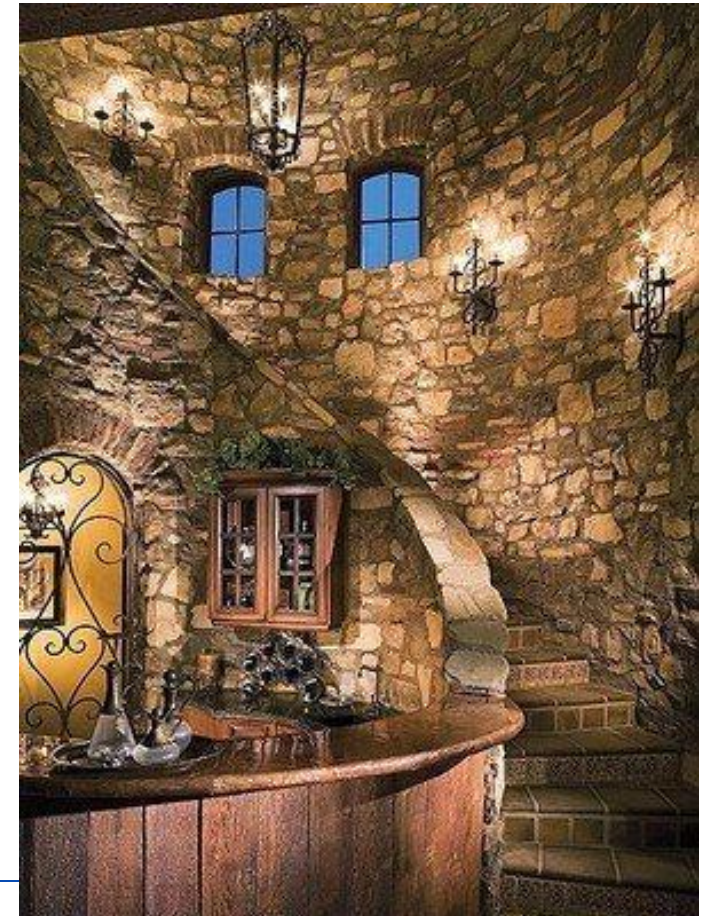
```
def sample(count):
    sendPin = 23
    recievePin = 24
    gpio.mode(sendPin,"out")
    gpio.mode(recievePin,"in")
    gpio.write(sendPin,0)
    total = 0    # set low
    for x in xrange(1,count):
        time.sleep(0.01)
        gpio.write(sendPin,gpio.HIGH)
        while(gpio.read(recievePin) == False):
            total += 1
        gpio.write(sendPin,gpio.LOW)
    return total
```

Contents

| | |
|---|--|
| 1 | Experiment: Button |
| 2 | Experiment: Microswitch |
| 3 | Potentiometer (Variable Resistor, Pot) |
| 4 | Sense Touch Without Touch (Capacitive Touch Sensor QT113) |
| 5 | Experiment: Feel the Pressure (FlexiForce) |
| 6 | Experiment: Build Your Own Touch Sensor |
| 7 | Test Project: Haunted Ringing Bell |

Test Project: Haunted Ringing Bell

- You enter a silent-looking reception desk.
- There is nobody in sight to check you in.
- You reach for the bell on the desk (Figure 5-22),
- but before you touch it... it rings!



Test Project: Haunted Ringing Bell

- In the Haunted Ringing Bell project, you'll learn how to:
 - ❑ Build a gadget that reacts to your hand before you touch it.
 - ❑ Make things move with servo motors.
 - ❑ Control servo motors.
 - ❑ Package a project to look like an innocent, everyday object.



Figure 5-22. Haunted bell

Servo Motors

- A servo is a motor you can precisely control (Figure 5-23). You can tell a standard servo to turn to a specific angle, such as 90 degrees. There are also full rotation servos where you control just the direction and rotation speed.
- If you ever think you need a motor for your project, consider a servo first. The servo is in contrast to common DC motors, which are difficult to control and require extra components just for changing direction. Most moving things in Arduino and Raspberry Pi prototyping projects are done with servos.

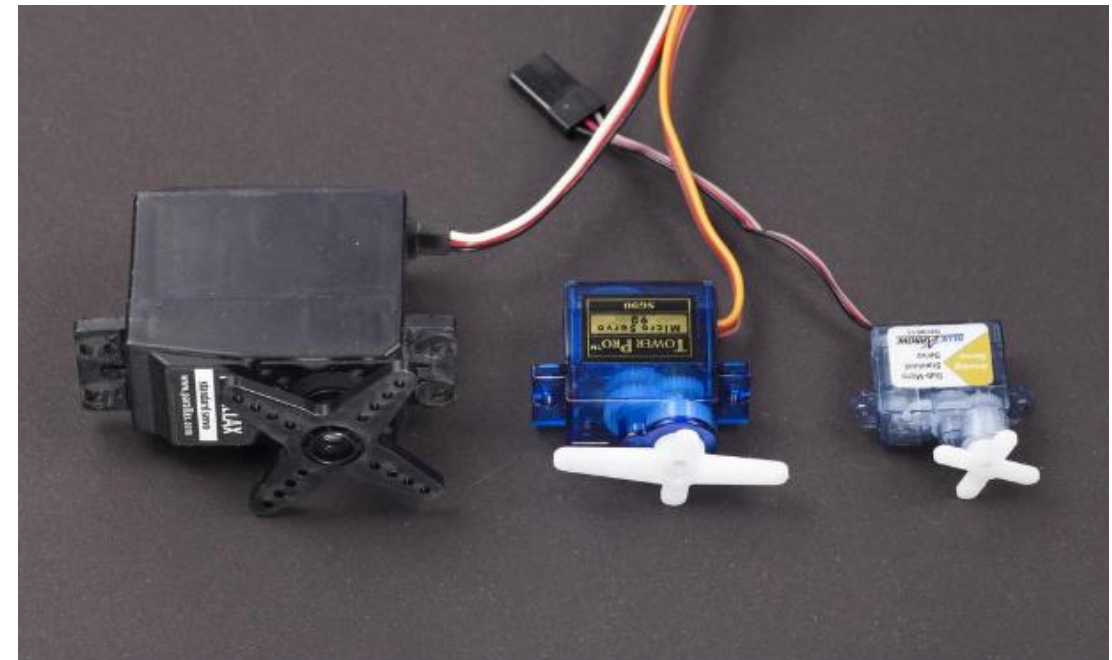
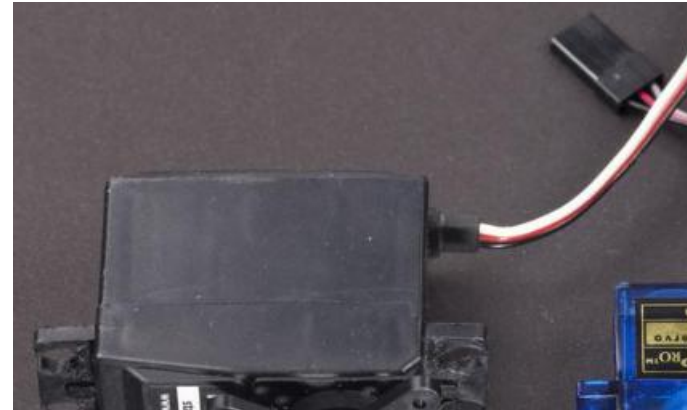


Figure 5-23. Different servos

Servo Motors

- A servo has three wires: black ground (0 V), red positive (+5 V), and control (yellow or white).



- Arduino sends a continuous stream of pulses to control wire. The length of these pulses tell the servo which angle to move to. The servo turns to this angle and then maintains it as long as pulses keep coming from Arduino.

Servo Motors

- The square wave (the pulse) is easy to create by rapidly changing a digital output pin between LOW and HIGH. To control the servo, Arduino must send 50 pulses a second, in other words, at the rate of 50Hz.
- The pulses are about 1 ms to 2 ms long. The longer the pulse, the bigger the angle. The pulse length between minimum and maximum angle centers the servo. Often, the center is about 1.5 ms.
- For most servos, it's not easy to find data sheets. But the update frequency (pulses per second) is usually the same, and it does not have to be exact.

Servo Motors

- The actual pulse length varies between servos, but it's easy to find it out experimentally. To get an idea of pulse length versus angle, see the example values in Table 5-1.

Table 5-1. Pulse length controls servo angle, example values

| 脉冲长度ms | 角度 | 说明 |
|--------|---------------|--------------------|
| 0.5 ms | $< -90^\circ$ | 旋转角度超过范围，齿轮发出怪怪的声音 |
| 1 ms | -90° | 最左边 |
| 1.5 ms | 0° | 中间 |
| 2 ms | 90° | 最右边 |
| 2.5 ms | $> 90^\circ$ | 超过范围，齿轮发出怪怪的声音 |

Finding Servo Range

- You can find out the servo range experimentally. This code is essentially the “Hello World” for servos.

| 脉冲长度μs | 角度 | 说明 |
|--------|-------|----------------|
| | -90 ° | 最左边，使用代码进行测试 |
| | 0 ° | 中间，最左边和最右边的平均数 |
| | 90 ° | 最右边，使用代码进行测试 |

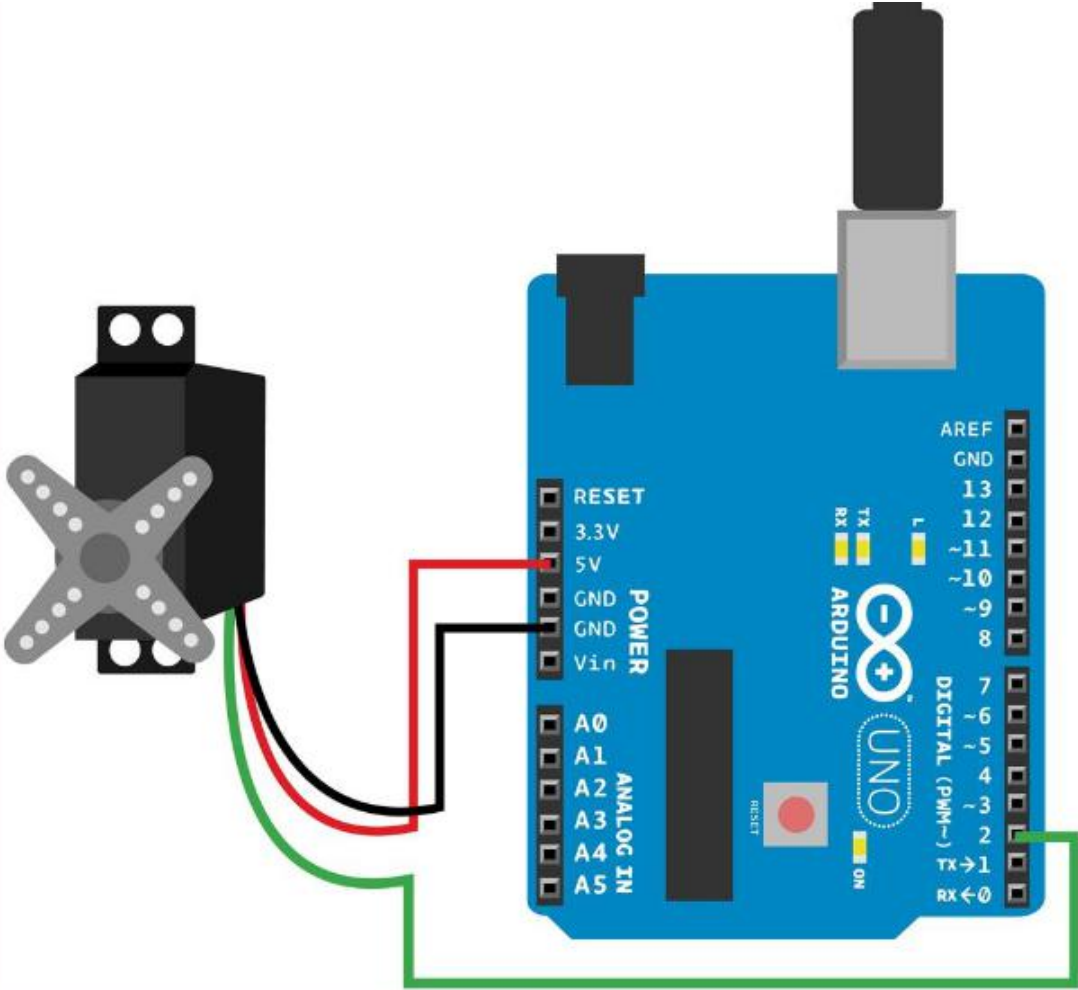


Figure 5-24. Servo connected to D2 for servo_range.ino


Finding Servo Range

Example 5-13. servo_range.ino

// servo_range.ino - turn servo and print values to serial

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

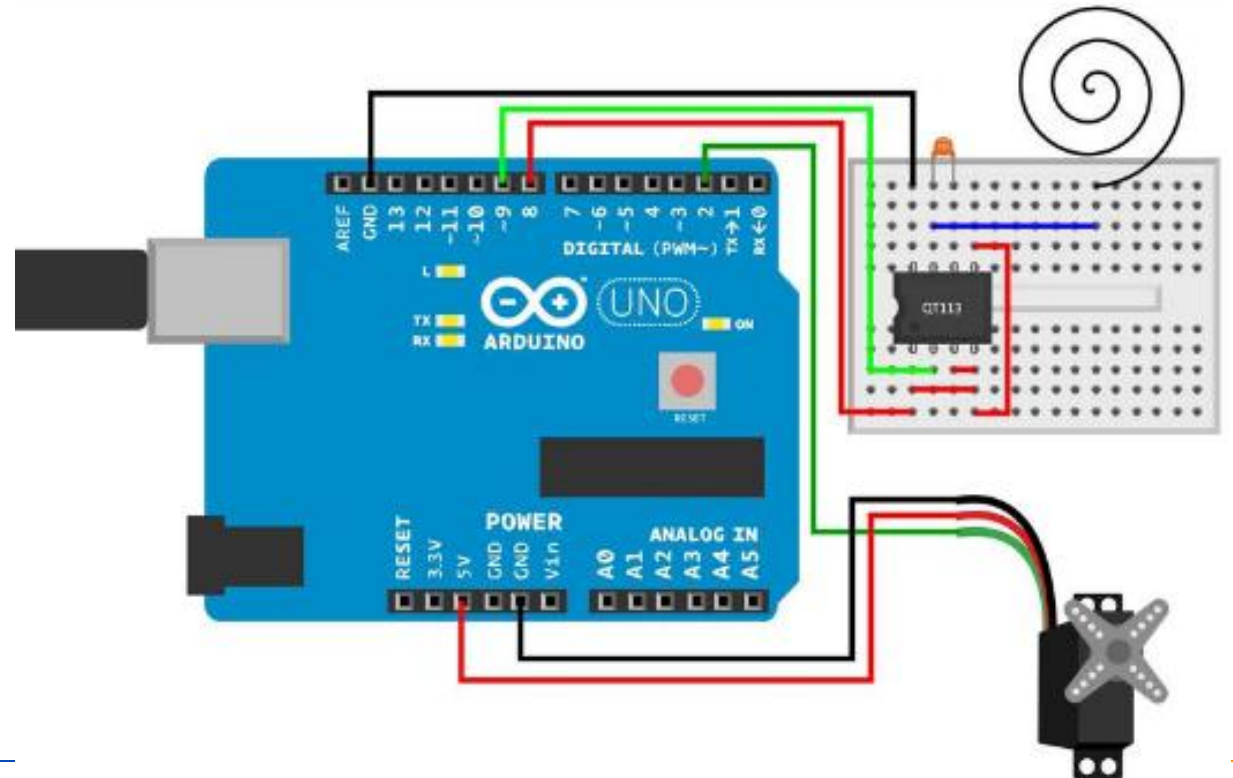
```
int servoPin=2;
void pulseServo(int servoPin, int pulseLenUs) { //❶
    digitalWrite(servoPin, HIGH); //❷
    delayMicroseconds(pulseLenUs); //❸
    digitalWrite(servoPin, LOW); //❹
    delay(15); //❺
}
void setup(){
    pinMode(servoPin, OUTPUT);
    Serial.begin(115200);
}
```



```
void loop(){
    for (int i=500; i<=3000; i=i+2) { //❻
        pulseServo(servoPin, i); //❼
        Serial.println(i);
    }
}
```

Haunted Bell Code and Connection for Arduino

- Connect the servo motor and capacitive touch sensor as in the earlier examples (see Figure 5-25). Again you'll need to use an 10-500 nF capacitor with QT113. The best value for us was 300 nF.



Example 5-14. haunted_bell.ino

Example 5-14. haunted_bell.ino

// haunted_bell.ino - bell rings just before you touch it

// (c) BotBook.com - Karvinen, Karvinen, Valtokari

```
int servoPin=2;
int sensorPin = 9;
int sensorPowerPin = 8;
int hasRang = 0;          //❶

void pulseServo(int servoPin, int pulseLenUs) { //❷
    digitalWrite(servoPin, HIGH);
    delayMicroseconds(pulseLenUs);
    digitalWrite(servoPin, LOW);
    delay(20);
}
```

```
void cling() { // ❸
    for (int i=0; i<=3; i++) { //❹
        pulseServo(servoPin, 2000);
    }
    for (int i=0; i<=100; i++) {
        pulseServo(servoPin, 1000);
    }
}

void setup(){
    pinMode(servoPin, OUTPUT);
    pinMode(sensorPowerPin, OUTPUT);
    digitalWrite(sensorPowerPin,HIGH);
    pinMode(sensorPin,INPUT);
}
```

Example 5-14. haunted_bell.ino

```
void loop(){
    int touch = digitalRead(sensorPin);    // ⑤
    if(touch == HIGH) {                    // ⑥
        hasRang = 0;
    }
    if(touch == LOW && hasRang == 0) { // ⑦
        cling();                           // ⑧
        hasRang = 1;                        // ⑨
        digitalWrite(sensorPowerPin,LOW);
        delay(1);
        digitalWrite(sensorPowerPin,HIGH);
    }
    delay(100);
}
```


Attaching Servo to Ringing Bell

- Use hot glue to attach the servo inside the ringing bell (Figure 5-26). Before gluing, make sure that the servo arm pushes the moving part inside the bell so that it gives a solid sound. Servo movement should also allow the bell button to move naturally. It might take a few tries to get the servo attached exactly the right way. Now your haunted ringing bell is ready to spook unsuspecting victims.



Figure 5-26. Servo inside the bell

Chapter 5. Touch

- You have now seen many kinds of touch and near-touch. You've played with buttons, microswitches, touch switches, and even used touch sensors without touch. After you've let the bell haunt your friends for a while, you can start applying this knowledge to your own projects. Think of any electronic devices around you: most sense touch in one way or another.