C4DI Hardware Group
Theremin

Vsn. 1.0 Rob Miles 2018

Introduction

Welcome to our Arduino hardware sessions.

Please follow the instructions carefully. If you get the wiring wrong your programs will not work and there is a good chance that you will destroy the delicate circuitry in the device that you are using.

In this session you will learn a bit about electronics and how to control simple circuits using an Arduino device. Here are a few conventions used in the text.

This indicates a warning to be careful about this bit. If you get it wrong it might be time to buy a new device.

This indicates an activity you should perform in at this point in the text. You may be given precise instructions, or you may have to work something out for yourself.

This indicates something that you may want to think about later.

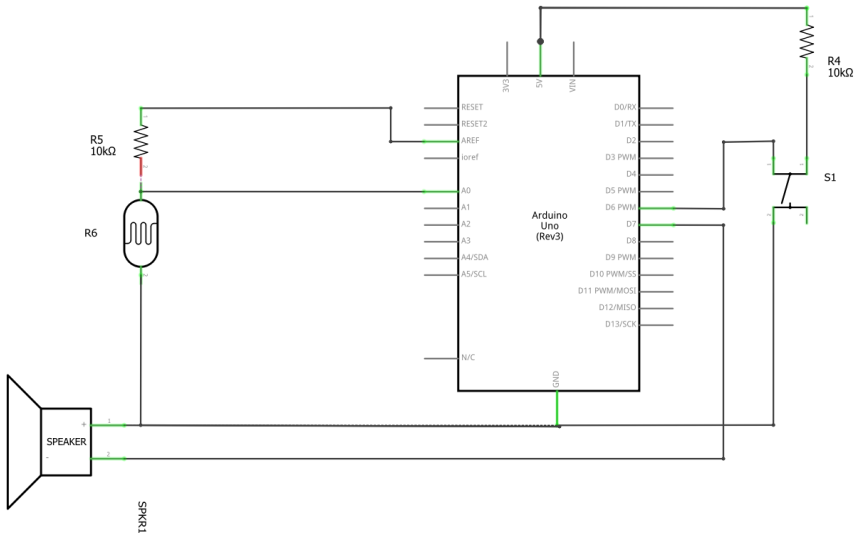
There are two parts to this work. We have to make the circuit (build the hardware) and then we have to create a program to use the devices (write the software).

The Problem

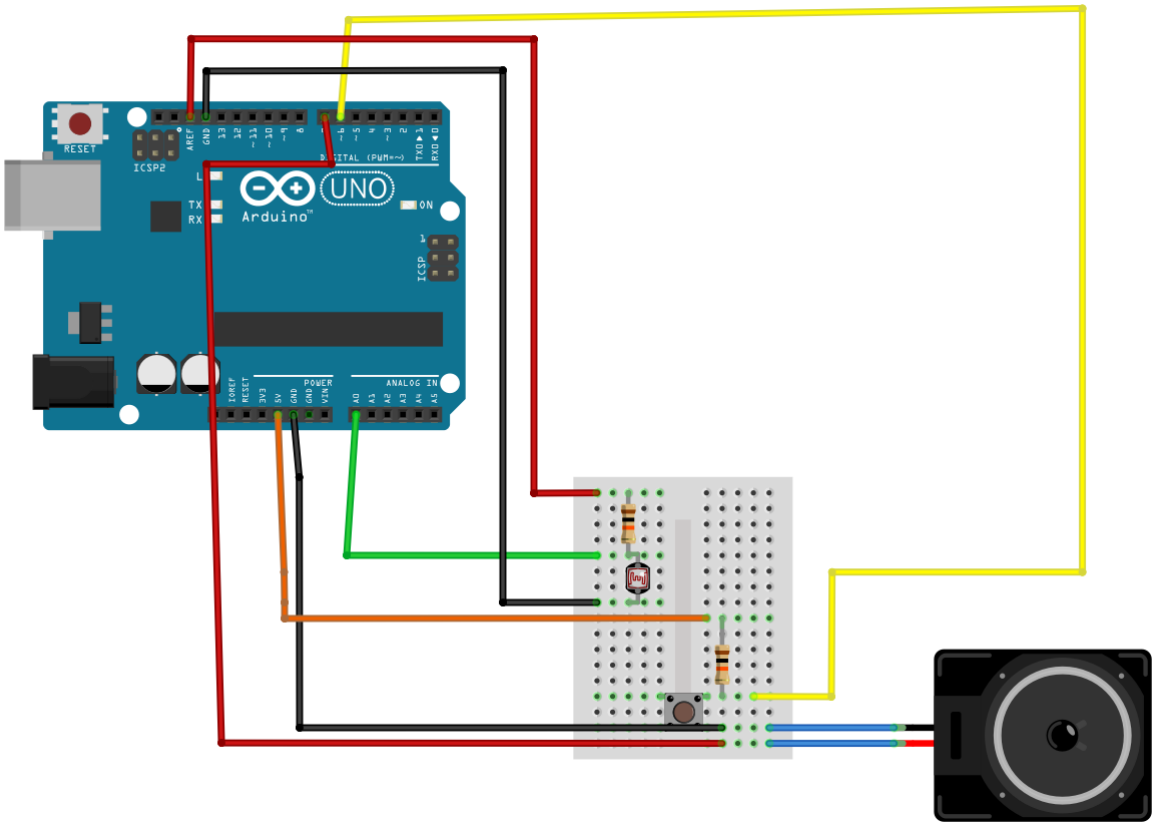
We are going to make a kind of musical instrument, a Theremin. This is an instrument that you play by moving your hands closer and further away from the instrument. It's used for spooky sound effects in science fiction movies. We are not going to use distance for our version, we are going to use light, building on the light sensor we made last time and adding a speaker. Then we are going to play with other sound features to make interesting sounds from our speaker.

Arduino Hardware

We can use a digital port from the Arduino to directly drive the speaker. We also need an analogue input that the program can read to allow the player to change the sound and we also need a button that we can press to trigger sound playback.



This is the circuit diagram that should do all this. The light sensor is connected to A0, the sounder is connected to D7 and the button is connected to D6.



This shows how the components are connected.

Create the circuit on your breadboard.

Arduino Software

The Ardunio library provides a function called tone which will play a tone for a particular length of time. The program selects the pin to use, the frequency of the tone and the number of milliseconds duration. We can use this to produce sounds:

```
int lightPin = 0;
int speakerPin = 7;
int buttonPin = 6;

void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop()
{
  int val = analogRead(lightPin);
  if(!digitalRead(buttonPin))
  {
    int freq = val;
    int length = 30;
    int gap = 20;
    tone(speakerPin, freq, length);
    delay(gap);
  }
}
```

The program above sets up the input and output devices and then repeatedly reads the analogue input to get a value that is then used to control the frequency of the note that is played by the tone command. Note that the tone is only played when the button is pressed.

Enter the program and run it. Save it in a sketch called “Music”. You should hear sounds when you press the button. If you adjust the rotary control you should be able to change the frequency of the sound.

You may find that the sound is not very good. Perhaps only a very limited range of light values is making a sound.

How do you fix this?

Perhaps we can modify the value from the sensor to make a more interesting range of frequencies. We could subtract a base value from the sensor reading and then multiply the result by a factor to make things sound better.

Making more Interesting Sounds

We can make more interesting sounds by considering how the tone function is used:

```
int freq = 1000;
int length = 100;
tone(speakerPin, freq,length);
delay(length+1);
```

The first parameter is the `speakerPin`, this identifies the output to be produced. The second is the frequency. The third is the length.

You might be wondering about the reason for the call of the `delay` method after we have started the tone playing. This is because the `tone` method is rather clever. It plays the sound in the background, while our program continues running. We want a given note to play before we change the frequency.

Note that this is a lovely example of a "kludge", in that from a programming point of view we shouldn't need this extra millisecond, but from a "making it work properly" perspective we definitely do.

If you are not convinced, try removing the delay and see what happens.

We can make a more interesting sound by playing a sequence of tones of different frequencies:

```
if (!newButtonState)
{
  // Button has just been pressed
  int freq;
  int length = 30;
  for ( freq=100; freq < 2000; freq=freq+200)
  {
    tone(speakerPin, freq,length);
    delay(length+1);
  }
}
```

This code replaces the simple tone generation with a for loop that takes the frequency from 100 to 2000 in steps of 200.

See if you can find a way to triggering this kind of whoosh by using the light sensor.

Adding Randomness

You can make sounds much more interesting by adding a little randomness to them. The Arduino library has a random function that will return a random number in a given range.

```
int dialPin = 0;
int speakerPin = 7;
int buttonPin = 6;
boolean oldButtonState;

void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
  oldButtonState = digitalRead(buttonPin);
}

void loop()
{
  boolean newButtonState = digitalRead(buttonPin);
  if(oldButtonState != newButtonState)
  {
    // Button has changed state

    // Update the old button state
    oldButtonState = newButtonState;

    // See if the button is now down
    if (!newButtonState)
    {
      // Button has just been pressed
      int freq;
      int length = 30;
      int i;
      for ( i=0; i < 50; i=i+1)
      {
        freq = random(1000,2000);
        length = random(30,40);
        tone(speakerPin, freq,length);
        delay(length+1);
      }
    }
  }
}
```

This program uses the random function to play sounds of different frequencies and lengths.

Start a new Arduino sketch called “Disrupter” and put the above program in it. You can vary the high and the low frequencies to get different sound effects. Throw in some light sensor values to make things even more interesting.

Rob Miles 2018