

Raspberry Pi GPIO – Musical Floppy Drive



Before I start this Project Tutorial I want to give credit where it is due.

Thank you to Scott Vincent – the writer of the C Source Code for the StarWars Theme song – Thanks for helping me resolve some of my problems as I was completely new to C – and Thanks for putting the code out there for others to use.

I also want to thank Gordon at www.wiringPi.com (drogon) – the writer of WiringPi for taking the time to help me out with the software – and also for putting it out there for people like me to use. Before wiringPi I was using RPi.GPIO in Python – and it was great for learning – but Limited with it's capabilities. WiringPi has really opened up a world of GPIO possibilities for me.

CONTENTS

- **Quick Setup Steps (Pin Wiring / Software)**
- **DO's and DONT's (Things will save time – or cause problems)**
- **How it works (Briefly)**
- **Parts List (What you will need – what I used)**
- **Preparing the Hardware**
- **Powering the Floppy Drive**
- **How to Wire everything up (Floppy pins / Gpio pins / Power)**
- **Software Set-up – And usage**
- **Test & Configure GPIO (Before running song)**
- **Using a Speaker to test the GPIO Output**
- **Running the Song (edit, compile, run in C)**
- **Source Code for the Starwars Theme – in C**
- **Using different GPIO Pins**

QUICK SETUP STEPS

Your Raspberry Pi should come with wiringPi as default – assuming you are running Raspbian. If not it is easy to install...see below.

Pin Wiring:

GPIO pin 0 -----> Floppy Pin 18

GPIO pin 1 -----> Floppy Pin 20

Jumper between Floppy Pin 11 and Floppy Pin 12

5v DC on the Floppy power Pins

0v (-) on the Floppy Casing (negative from 5V supply)

Software:

Make sure your RPi is up to date for the new software

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

Install git – so you can download the software

```
sudo apt-get install git-core
```

Clone / Copy the wiringPi software

```
git clone git://git.drogon.net/wiringPi
```

Navigate into the new wiringPi folder

```
cd wiringPi
```

Build the program

```
./build (may need to do sudo ./build)
```

Copy the C code at the very bottom (or from the link directly)

Then create and open a new file for editing

```
nano floppymusic.c
```

Paste the copied code into the new file

Save by hitting CTRL+X, Y, Enter

Compile the C Program (song) ready to be executed

```
gcc -o floppymusic -std=c99 -lwiringPi floppymusic.c
```

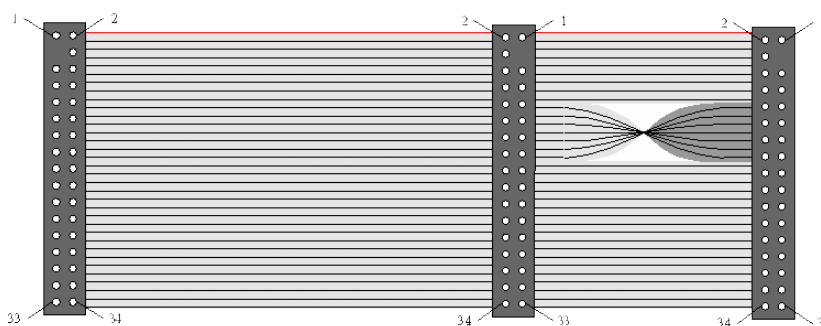
Run the song program

```
sudo ./floppymusic
```

This method just skips ahead to running the StarWars theme song straight away - there are methods listed in the detailed version (below) that allow the user to configure the pins are responding to the software by simply switching them between high and low first...this is recommended because you can determine that the GPIO pins are set up correctly.

DO's AND DONT's FOR THIS PROJECT >>> PLEASE READ FIRST

1. Floppy Drive Cables...with a twist
2. GPIO Pin numbers....some have different layouts

1.Floppy Disk Drive cables with a twist:

Using the grey IDE looking cable for convenience? Look at the diagram above. Most will have this twist – so beware of using the wrong pins because of this. Below I will explain how you can overcome this easily.

Having the correct floppy-to-motherboard cable is very convenient, in my case my jumper wires plugged nicely into the black connector – giving me easy and direct access to floppy drive pins.

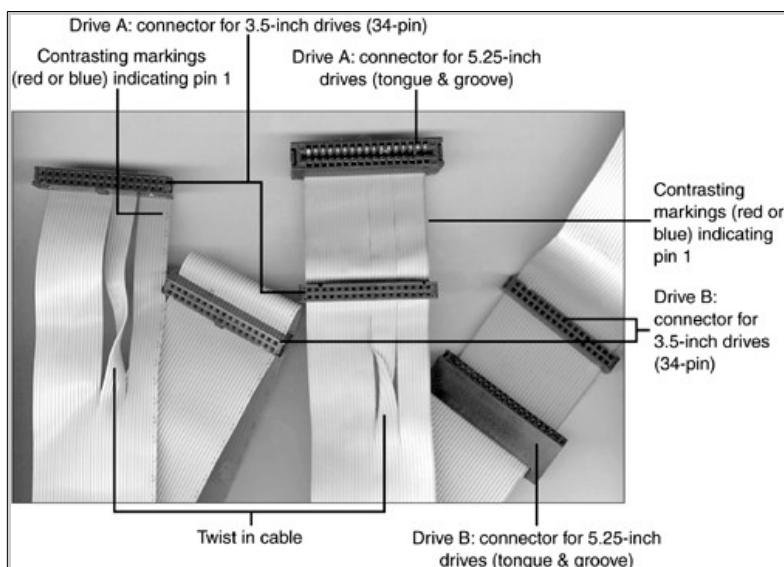
Beware that most have a twist in them and this will cross wire pins to the wrong ones – which could result in unwanted problems.

I had 3 floppy drives. They had the grey cables included. These cables originally connected from the floppy drive to motherboard, and have a black connector both ends, that fits across ALL the pins on the back of the drive.

NOTE most of these cables have a twist in them, so some of the wires are manufactured to be crossed over – this will cause you problems. If your cable has the twist you have 2 options (if you want to use the cable still)

1. Use the pinout chart to identify which pins are being crossed over and adjust your project to use the right pins in different locations
2. Cut the cable at the twist...untwist...strip the ends of the wires/pins you will be using on the cable....tape up the unused ones for safety.

Use this reference diagram to see the different types of Floppy Drive cables:



HOW IT WORKS (BRIEFLY)

The non-technical explanation:

You have heard a floppy drive, or a hard drive, or even a printer making noise as it performs tasks at your request. This is basically what we are going to make the floppy drive do – but at certain tones.

The [wiringPi](#) software for Raspbian allows the user to interact with the GPIO pins a lot easier than [Rpi.GPIO](#) and also allows a wider range of things to be done with them, without the need for admin or sudo access each time.

Previously you could just switch the GPIO pins between High and Low - or - On and Off. You could then write a Python program to set things to blink at certain times. Popular basic GPIO projects were things like the LED blinking circuit, and button activity – showing a message when a button was pressed.

Wiring Pi lets you send signals at Specific Frequencies to your GPIO Pins

A signal generated at a specific frequency will make a speaker or motor move at a specific speed – or make a specific tone. If you change the frequency, you change the tone or speed it vibrates.

So a complex program in C has been written, organising ALL of the right frequencies into chords and timing signatures – to create a tune.

When you have mastered this, there are other programs that allow midi to code conversions – allowing you to play any midi song, and there is also software for Raspbian that allow 6 channel audio playback....so you could split a song into 6 channels and make 6 floppies play a song – each doing their own task – or each one replicating a different instrument. This is the process behind the very popular Floppy Orchestra / Robot Orchestra Videos.

PREPARING THE HARDWARE

This can be very easy if you have the right sort of jumper wires and connectors to connect everything – or a nightmare if you don't. I mess with electronics and happened to have a load of 2 and 3 pin connectors that I cut off old circuit boards and appliances – as-well as an old and fried ATX PC Power supply with Floppy Connectors ready to cut off. I also have a GPIO Breadboard / cobbler Kit that came with my Raspberry Pi – it allows a cable (like an IDE cable) to connect to the GPIO pins easily and also to a breadboard – so I have safe and easy access to ALL my GPIO pins – making life a lot easier.

I bought a pack of Male Jumper wires . I got a pack of 50 jumpers on Amazon UK for under £5 and I have seen the little breadboards for also under £5 – a very good investment if you plan to do more electronics projects with your Raspberry Pi.

If uou don't have all of these don't give up!

Either get creative with your connections (BE CAREFUL not to touch the wrong pins – do this stuff without your Pi connected – until you are sure you are ready to proceed) Or – there is another easy way. If you can find an OLD Computer tower you may be in luck – there are a load of Female jumper wires going from the front case buttons (power/ reset/ indicator LED's) directly to the motherboard. These will fit perfectly over ALL pins in this project.

You could use these to connect GPIO pins to floppy drive pins.....and also for the + and – Floppy Power pins. Your Old PC tower may have a power supply inside with all the wiring attached...if it does, look for a 4 pin small white connector.

Either Cut it off with enough wire to play with.....or get that power supply out of the PC tower and use it to power your floppy directly(you will need a kettle-like lead).

Your Floppy drive doesn't need a floppy disk for this. And you don't need to remove the casing. Although it can be useful to remove the top case to see the motor – when configuring your GPIO pins with the software you will be able to see the minute movements in the motor

WHAT YOU WILL NEED ALTOGETHER:

Raspberry Pi running Raspbian

GPIO Access (either a breakout/cobbler kit or some decent female jumper wires)

Floppy Drive (just ONE floppy to play this song)

Power supply for your Floppy Drive * [See Below](#)

Basic wire stripping tools – and I used connector blocks for connections

* Floppy power supply

You have a few options here – but the floppies have odd requirements when it comes to grounding. If you had an old PC and managed to get a working ATX power supply with floppy connectors from it – you are lucky and I recommend using it. I didn't do it that way. I powered BOTH the Floppy and Raspberry Pi off the same 5V aspberry Pi adapter and used a 5vDC phone charger adapter to provide JUST a (-) negative grounding to the floppy casing. I was able to do this because my GPIO

breakout board allows me to use a 5v DC + and – separately to the GPIO pins for this project – so I am lucky. However...this is only enough to power the floppy drive...it needs heavier grounding to get it singing.

I used another 5v Phone charger adapter – and attached the GROUND / NEGATIVE to the floppies outer casing (metal)

I have a 5v DC phone charger plug – with the end cut-off and stripped back a bit. This will power a floppy instead of the Pi but you will still need to ground the outer casing of the floppy to the Pi's GPIO ground (-0v) if you do it this way – You will see a few GPIO pins are used for GND or Ground.

POWERING THE FLOPPY DRIVE

Which method you use will depend entirely on your circumstances, and what you have available to use.

If you have the Cobbler/Breakout Breadboard for the GPIO pins – you can power the Floppy drive with the RaspberryPi's 5v DC (+) and (-) >>> BUT...you will then need to Ground the Floppy Drive Casing with another power supply. I found that it can't ALL be done via the Pi. Either you power the floppy with the pi and ground the casing with another supply...or power the floppy with another 5v supply and ground it to the Pi.

You have a few options – but the floppies have odd requirements when it comes to grounding. If you had an old PC and managed to get a working ATX power supply with floppy connectors from it – you are lucky and I recommend using it to power the floppy drive...then just ground the casing to one of the Pi's GND pins.

I didn't do it that way. I powered BOTH the Floppy and Raspberry Pi off the same 5V aspberry Pi adapter and used a 5vDC phone charger adapter to provide JUST a (-) negative grounding to the floppy casing. I was able to do this because my GPIO breakout board allows me to use a 5v DC + and – separately to the GPIO pins for this project – so I am lucky. However...this is only enough to power the floppy drive...it needs heavier grounding to get it singing.

I have a 5v DC phone charger plug – with the end cut-off and stripped back a bit. This will power a floppy instead of the Pi but you will still need to ground the outer casing of the floppy to the Pi's GPIO ground (-0v) if you do it this way – You will see a few GPIO pins are used for GND or Ground.

HOW TO WIRE IT ALL UP

If you have everything ready – your connectors and jumper wires...and you have a power supply for your Floppy Drive, you are ready to wire it all up.

NOTE: THIS IS WHAT WORKED FOR ME: There are a few RPi tutorials for this online – some look dead simple. I was messing about for almost 3 days with this and the way I am explaining Works for Me – I tried every possible way of grounding it and powering it – other than this method all I could do was power the Floppy LED with the GPIO signals. I had 3 Floppy Disk Drives and only 1 out of 3 worked for this project.

GPIO pin 0 -----> Floppy Pin 18

GPIO pin 1 -----> Floppy Pin 20

Jumper between Floppy Pin 11 and Floppy Pin 12

5v DC on the Floppy power Pins

0v (-) on the Floppy Casing (negative from separate 5V supply – or the Pi)









































See the Floppy Drive Pin number chart to identify the correct pins:

STANDARD FLOPPY DRIVE PIN-OUT CHART

Floppy Drive Port - Internal

1	-	Not Connected	2	-	Not Connected
3	-	+ 5 Volts	4	-	Not Connected
5	-	+ 5 Volts	6	-	Not Connected
7	-	+ 5 Volts	8	-	Index
9	-	+ 5 Volts	10	-	Drive 0 Select
11	-	+ 5 Volts	12	-	Drive 1 Select
13	-	Ground	14	-	Not Connected
15	-	Ground	16	-	Motor On
17	-	Ground	18	-	Direction
19	-	Ground	20	-	Step
21	-	Ground	22	-	Write Data
23	-	Ground	24	-	Write Enable
25	-	Ground	26	-	Track 0
27	-	Ground	28	-	Write Protect
29	-	+ 12 Volts	30	-	Read Data
31	-	+ 12 Volts	32	-	Disk Side Select
33	-	+ 12 Volts	34	-	Disk Change

NOTE: Power is supply on the floppy drive controller cable.
These lines are normally ground lines.

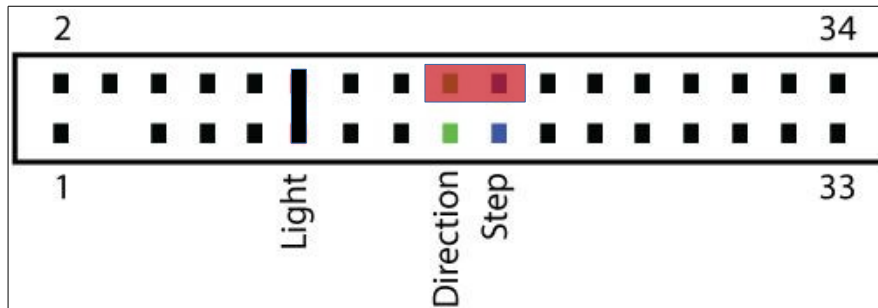
Raspberry Pi 2 Model B (J8 Header)						
GPIO#	NAME			NAME	GPIO#	
	3.3 VDC Power	1			2	5.0 VDC Power
8	GPIO 8 SDA1 (I2C)	3			4	5.0 VDC Power
9	GPIO 9 SCL1 (I2C)	5			6	Ground
7	GPIO 7 GPCLK0	7			8	GPIO 15 TxD (UART) 15
	Ground	9			10	GPIO 16 RxD (UART) 16
0	GPIO 0	11			12	GPIO 1 PCM_CLK/PWM0 1
2	GPIO 2	13			14	Ground
3	GPIO 3	15			16	GPIO 4 4
	3.3 VDC Power	17			18	GPIO 5 5
12	GPIO 12 MOSI (SPI)	19			20	Ground
13	GPIO 13 MISO (SPI)	21			22	GPIO 6 6
14	GPIO 14 SCLK (SPI)	23			24	GPIO 10 CE0 (SPI) 10
	Ground	25			26	GPIO 11 CE1 (SPI) 11
30	SDA0 (I2C ID EEPROM)	27			28	SCL0 (I2C ID EEPROM) 31
21	GPIO 21 GPCLK1	29			30	Ground
22	GPIO 22 GPCLK2	31			32	GPIO 26 PWM0 26
23	GPIO 23 PWM1	33			34	Ground
24	GPIO 24 PCM_FS/PWM1	35			36	GPIO 27 27
25	GPIO 25	37			38	GPIO 28 PCM_DIN 28
	Ground	39			40	GPIO 29 PCM_DOUT 29

Attention! The GPIO pin numbering used in this diagram is intended for use with WiringPi / Pi4J. This pin numbering is not the raw Broadcom GPIO pin numbers.

<http://www.pi4j.com>

GPIO Pins 0 and 1 are small numbers 11 and 12 (11 and 12 as you count them on the pi)

FLOPPY DRIVE PIN NUMBERING DIAGRAM
(Light pins 11 and 12 are connected together)
(Highlighted in red – 18 and 20 are to GPIO)



Hardware Step 1: Connect Pins 11 and 12 on the floppy – to each other.

Hardware Step 2: Power the Floppy Drive with 5v DC

Hardware Step 3: Ground the casing of the Floppy Drive

Hardware Step 4: Connect GPIO Pins 0 and 1 to the Floppy Pins 18 and 20

In general the pins are laid out as follows....as you look at the floppy pins ALL even pins are the TOP row. **ALL ODD** numbered pins are the **BOTTOM** row. The grounds are always on the ODD / bottom row. The numbers start at the **BOTTOM LEFT** with **NUMBER 1** > then above it or **TOP LEFT** is **NUMBER 2**

Hardware Step 1:

Notice on the previous diagram, 2 of the pins are coloured in **RED** and are labeled “**Light**”. You need to connect these 2 pins with a jumper – to allow the circuitry to work – when you have your floppy drive hooked up to the power, you will see the LED on the front light up if you have this jumper in place

I recommend using a real jumper, but you can get creative...I had some 2 pin connectors off old circuit boards, I put one over the 2 pins, and connected the wires together. On OLD hard drives...the IDE cable type, a lot of them had a second smaller set of pins...these used to have a jumper that would allow the drive to be switched between **Master**, or **Slave**. One of these jumpers would be perfect.

Bottom line is – you need to connect those 2 pins...pin 11 and 12 – the 6th row from the left – as shown in red on the diagram above.

Hardware Step 2:

I covered power above...see [POWERING THE FLOPPY DRIVE](#) but here's what I have found. There are a few ways to do this, but the floppy is pretty fussy when it comes to grounding. Ideally you want the floppy and Pi to be grounded together one way or another.

Which method you use will depend entirely on your circumstances, and what you have available to use.

If you have the Cobbler/Breakout Breadboard for the GPIO pins – you can power the Floppy drive with the RaspberryPi's 5v DC (+) and (-) >>> BUT...you will then need to Ground the Floppy Drive Casing.

When you have supplied the floppy drive with power, the LED will light up on the front..if you have connected floppy pins 11 and 12 in the previous step.

If you are looking at the back of the floppy drive, with the drive the right way up (floppy Data Pins should be positioned with 1 at the bottom left) then the power connector pins from left to right are as follows:

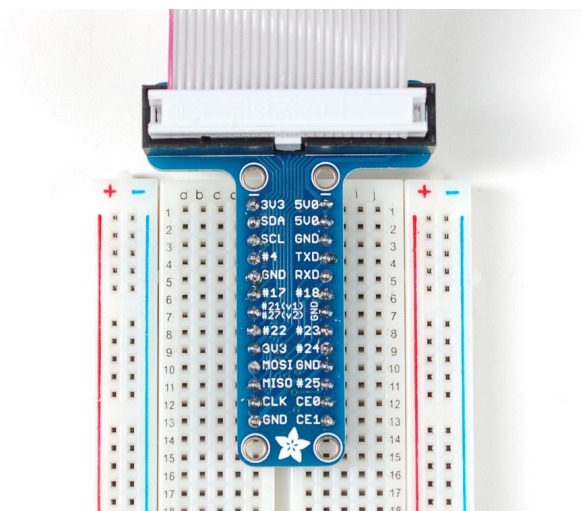
Pin 1 (Left Pin)	Not Used
Pin 2 (Middle-Left)	Not Used
Pin 3 (Middle-Right)	Negative 5v DC (-) - BLACK WIRE
Pin 4 (Right Pin)	Positive 5v DC (+) - RED WIRE

Hardware Step 3:

Hardware Step 3:

Hardware Step 4:

Connect GPIO Pins 0 and 1 to the Floppy Pins 18 and 20. This is easy if you have a breadboard, or cobbler kit for your Pi – Mine looks like this but with direct power from the Pi to the left and right power rows:



This picture is for an older Raspberry Pi with less GPIO Pins. A Raspberry Pi 2 will have 40 GPIO Pins and a slightly bigger connector. These kits normally come with the proper jumper wires, that click nicely into the breadboard – and into most female connectors including the floppy IDE cable. Mine has 2 extra features....the power strips on the breadboard are on the very left and very right – when you put a + and – into any slot on these rows the WHOLE row becomes available as + and -

My Pi Connector has a 5v DC and 3v DC's that go into the power slots on the left and right of the breadboard – giving me seperate power supply without using the GPIO's needed for projects. It is VERY useful – this is what I power my floppy with.

With this kit I just put my floppy power supply wires into the 5v + and – on the right – where my GPIO connector gives it 5v DC

Then I take 2 jumper wires. Put them into GPIO 11 and 12 (GPIO 1 and GPIO 0) and put the other ends into the floppy pins 18 and 20 – Highlighted in red on the Floppy Drive Pin-out chart I included further up.

On the next page I cover the Software side of the project.

SOFTWARE SETUP AND USAGE

Install wiringPi:

(Raspbian comes with wiringPi as default, so you may already have it. There's no harm running the installation anyway, in case you aren't up to date with your version)

The commands I used to download and install wiringPi were taken from the wiringPi website:

www.wiringPi.com/download-and-install/

Or follow my steps below:

I did this project with a Raspberry Pi 2 (B) Running Raspbian
You need Git installed:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install git-core
```

To download the wiringPi software using GIT:

```
git clone git://git.drogon.net/wiringPi
./build
```

When my installation finished I was given a message saying I still need **-lwiringPi** and **-lwiringPiDev**. You won't need these – they are for compiling code within wiringPi.

And that should be all you need for installation of wiringPi. If you still have problems after running these installation commands I recommend visiting the wiringPi website www.wiringPi.com.

Test and Configure wiringPi is working with your GPIO Pins:

You will be testing your GPIO pins with some simple gpio commands. If you are following this guide start to finish you need to have the Floppy Drive wired up and ready.

If you just want to test your GPIO pins for now then you should make a simple circuit involving an LED and a resistor on the GPIO breadboard (my GPIO pins have a connector like the piWedge – it connects directly to a breadboard for easy access to the pins).

SEE DIAGRAM BELOW FOR GPIO PINS LAYOUT– I have included a diagram I found - It is actually made for wiringPi and the Raspberry Pi 2 – (you will find that different versions of Pi have different GPIO Pin numberings...and wiringPi has a numbering layout as-well.) THIS GUIDE is for a RASPBERRY PI 2 MODEL B – GPIO pins

When you have your Floppy pin 18 and Floppy pin 20 connected to GPIO 0 and GPIO 1 you can run some simple commands to test you are able to send signals to the pins

For now you are ONLY switching them between HIGH and LOW and you wont see a lot of activity while this is hooked to your floppy. I suggest having a simple LED and resistor circuit ready, connect your GPIO's 1 and 0 to the LED + resistor and watch it switch between high and low. Some say they hear a faint click from the floppy. I didn't.

Before you start playing around with wiringPi you may be interested to see just what it can do. In your terminal type:

```
man gpio
```

This will bring up a full manual of the commands and functions you can perform with wiringPi – as-well as how to use them correctly

TO TEST YOUR GPIO PINS ARE SYNCHRONISED WITH THE SOFTWARE

Type:

```
gpio mode 0 out
gpio mode 1 out
```

This prepares GPIO pins 0 and 1 – for **output** use.

Then type:

```
gpio write 0 0
gpio write 1 0
```

For me this switched BOTH pins to High (0 is high, 1 is low)....to make them low again:

```
gpio write 0 1
gpio write 1 1
```

the first number is your GPIO pin – the second is High or Low
I found that you can actually type

```
gpio write 0 on
gpio write 0 off
```

as-well! But I don't know how this will work on all systems.

Not all Floppies will react with the front LED to this pin switching. I had one that did, and others that didn't. If you want to make 100% certain your GPIO pins are working either use a digital multimeter set to DC Voltage of over 5V and look at how the pins react to you running the commands – or make a simple circuit as mentioned above...using an LED and appropriate resistor (voltage is about 5V you won't need a resistor to do a quick check but your LED won't last long like this...if you have no resistors just put a few LED's in series – they should be dimmer and share the load) At this point you ONLY want to confirm the commands are interacting with the GPIO Pins

The next step is to run the Star Wars Theme tune in C.

TESTING GPIO OUTPUT WITH A SPEAKER

A speaker will vibrate to the frequencies sent through the GPIO pins – and will play the Starwars Song very well.

I use a small speaker to test the output is working before hooking up the floppy drive.

I am also planning the more complex version using multiple drives – for now I will use speakers to simulate the floppy drives. This is not a way of playing music – the sound is the same buzzing you expect to hear from the floppy drive – and if you use a tube or cone over the speaker you can amplify the sound a little.

NOTE I am not suggesting to hook up a sub woofer and blast frequency music through your Pi. I used a VERY small piezo looking speaker at first and got bigger as I tested more. I found that bigger speakers provide better quality at less volume

Use a small speaker – One that worked well for me was as follows:

8 OHM 2 WATT

SIMPLY HOOK A SMALL SPEAKER TO THE SAME GPIO PINS (0 and 1) AND RUN THE SONG BELOW

Running the C Program with wiringPi:

If you are familiar to C this is easy stuff. But if like me you have never used C before and are familiar with low-level Python the following steps will be new to you. Heres why:

In Python you just write a code, save it as a .py file and execute it – either from a command terminal or from the code editor.

C is different and not as simple – although it isn't difficult either. I have done the hard work so you don't have to.

In the C language you have to:

- Write and save the program
- Compile it
- Make it executable
- Then Execute it

You have to compile the floppysong.c program and make it executable.

Firstly you have to make a file – put the C source code for the song in to it – There are 2 ways to do

this: Use the nano file editor from the terminal

Using a code editor like Geany or your own preference:

- Open the editor and start a new project / file
- Copy / Paste the source code into the new document
- Save it as floppymusic.c (Saving into the wiringPi folder)

The wiringPi folder was created upon install – it should be in your home folder in File Explorer

Using the In-Terminal editor Nano

- Navigate to wiringPi directory – `cd wiringPi`
- Type `nano floppymusic.c` (to create and open a file)
- Copy and paste the source code into the new file window
- Hit CTRL+X and then Y to save, then Enter to keep the name and return to terminal

NOW YOU HAVE THE SOURCE CODE SAVED APPROPRIATELY YOU NEED TO COMPILE THE PROGRAM:

At this point I want to mention that if you decide to use **different GPIO pins** to 0 and 1 (I blew my 0 and 1 pins so I ended up having to use 22 and 26 (which worked) and I can also safely control 24 and 27) you have to edit the top few lines of floppymusic.c code to display **YOUR** gpio pin number configuration – (it's just 2 numbers to change). Do this **BEFORE** you compile the code as you can't edit it afterwards (as far as I know).

If you are just going to use GPIO 0 and GPIO 1 – you don't have to change anything

When you returned to the terminal from nano you should be in the folder / directory called [wiringPi](#). From here you can **compile** your **floppymusic.c** program by typing:

```
gcc -o floppymusic -std=c99 -lwiringPi floppymusic.c
```

All you need to do to play the song from here is enter the following line of code (assuming your Floppy Drive is wired up correctly):

```
sudo ./floppymusic
```

You should get NO errors and the floppy will play the song. If you get Errors let me know, if not and it still doesn't work but it runs and you get no errors I suggest getting a SMALL speaker and wire it to the 2 x GPIO pins going to the floppy drive

If Like me you have to go back and use different pins, you'll have to start again with the source code – edit the Pin numbers in the top few lines of C code, save it in to the nano file again and compile it again:

```
gcc -o floppymusic -std=c99 -lwiringPi floppymusic.c
```

Ben Woodfield

raserppsprograms@gmail.com

09/10/2016

You should be getting music at this point....either through your small speaker or Floppy Drive.

STARWARS THEME SOURCE CODE IN C

>>> Written By Scott Vincent <<< and available directly from
www.raspberrypi.org/forums/viewtopic.php?t=69947&p=994442

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <wiringPi.h>

// pin 11 = wiringPi Pin 0. Use this for motor direction.
const int dirPin = 0;

// pin 12 supports pwm mode but it turns out I didn't need pwm mode in the end!
// pin 12 = wiringPi Pin 1. Use this for stepper motor.
const int stepPin = 1;

// Define an octave with naturals and sharps (Zz = rest)
enum { Cn, Cs, Dn, Ds, En, Fn, Fs, Gn, Gs, An, As, Bn, Zz };

// Define another one with flats and remaining sharps
enum { Bs, Df, Dn2, Ef, En2, Es, Gf, Gn2, Af, An2, Bf, Bn2, Zz2 };

/**
 * Frequencies in hundredths of Hz, e.g. middle A = 44000
 * 4 Octaves with 12 notes per octave, i.e. C to B
 */
const int freq[4][12] = {
    { 13081,13859,14683,15556,16481,17461,18500,19600,20765,22000,23308,24694 },
    { 26163,27718,29366,31113,32963,34923,36999,39200,41530,44000,46616,49388 },
    { 52325,55437,58733,62225,65925,69846,73999,78399,83061,88000,93233,98777 },
    { 104650,110873,117466,124451,131851,139691,147998,156798,166122,176000,186466,197553 }
};

/**
 * Frequency (in Hz) is converted to Floppy Delay using the formula:
 * 314000 / frequency = floppy delay
 * so middle A = 314000 / 440 = 714
 */
const int floppyConv = 31400000;

// Calculate all our floppy delays at the start
int floppyDelay[4][12];

// Song1 is the C major scale (note, octave, length)
const int song1_tempo = 120;
const int song1[][3] = {
    { Cn, 1, 1 },
    { Dn, 1, 1 },
    { En, 1, 1 },
    { Fn, 1, 1 },
    { Gn, 1, 1 },
    { An, 1, 1 },
    { Bn, 1, 1 },
    { Cn, 2, 1 },
    { -1, -1, -1 }
};

// Song2 is The Imperial March from Star Wars (note, octave, length)
const int song2_tempo = 104 * 8;
const int song2[][3] = {
    { Gn, 1, 8 }, // Bar 1
    { Gn, 1, 8 },
    { Gn, 1, 8 },
    { Ef, 1, 6 },
    { Bf, 1, 2 },

    { Gn, 1, 8 },
    { Ef, 1, 6 },
```

```
{ Bf, 1, 2 },
{ Gn, 1, 16 },

{ Dn, 2, 8 },
{ Dn, 2, 8 },
{ Dn, 2, 8 },
{ Ef, 2, 6 },
{ Bf, 1, 2 },

{ Gf, 1, 8 },          // Bar 4
{ Ef, 1, 6 },
{ Bf, 1, 2 },
{ Gn, 1, 16 },

{ Gn, 2, 8 },
{ Gn, 1, 6 },
{ Gn, 1, 2 },
{ Gn, 2, 8 },
{ Gf, 2, 6 },
{ Fn, 2, 2 },

{ En, 2, 2 },
{ Ds, 2, 2 },
{ En, 2, 4 },
{ Zz, 0, 4 },
{ Gs, 1, 4 },
{ Cs, 2, 8 },
{ Bs, 2, 6 },
{ Bn, 1, 2 },

{ Bf, 1, 2 },          // Bar 7
{ An, 1, 2 },
{ Bf, 1, 4 },
{ Zz, 0, 4 },
{ Ef, 1, 4 },
{ Gf, 1, 8 },
{ Ef, 1, 6 },
{ Gf, 1, 2 },

{ Bf, 1, 8 },
{ Gn, 1, 6 },
{ Bf, 1, 2 },
{ Dn, 2, 16 },

{ Gn, 2, 8 },
{ Gn, 1, 6 },
{ Gn, 1, 2 },
{ Gn, 2, 8 },
{ Gf, 2, 6 },
{ Fn, 2, 2 },

{ En, 2, 2 },          // Bar 10
{ Ds, 2, 2 },
{ En, 2, 4 },
{ Zz, 0, 4 },
{ Gs, 1, 4 },
{ Cs, 2, 8 },
{ Bs, 2, 6 },
{ Bn, 1, 2 },

{ Bf, 1, 2 },
{ An, 1, 2 },
{ Bf, 1, 4 },
{ Zz, 0, 4 },
{ Ef, 1, 4 },
{ Gf, 1, 8 },
{ Ef, 1, 6 },
{ Bf, 1, 2 },

{ Gn, 1, 8 },
{ Ef, 1, 6 },
{ Bf, 1, 2 },
{ Gn, 1, 16 },

{ -1, -1, -1 }
};
```

```
/**
 *
 */
static void resetMotor()
{
    // To reset head position move back 10 then forward 5
    digitalWrite(dirPin, LOW);
    for (int i=0; i < 10; i++){
        digitalWrite(stepPin, HIGH);
        digitalWrite(stepPin, LOW);
        delay(1);
    }

    digitalWrite(dirPin, HIGH);
    for (int i=0; i < 5; i++){
        digitalWrite(stepPin, HIGH);
        digitalWrite(stepPin, LOW);
        delay(1);
    }

    delay(400);
}

/**
 *
 */
static int init()
{
    if (wiringPiSetup() == -1){
        printf("Failed to initialize wiringPi\n");
        return 1;
    }

    pinMode(stepPin, OUTPUT);
    pinMode(dirPin, OUTPUT);

    resetMotor();

    for (int octave = 0; octave < 4; octave++){
        for (int note = 0; note < 12; note++){
            floppyDelay[octave][note] = floppyConv / freq[octave][note];
        }
    }

    return 0;
}

/**
 *
 */
static void playNote(int note, int octave, int length)
{
    static int dir = 1;
    int pause = floppyDelay[octave][note] * 10;

    int endTime = millis() + length;
    while (millis() < endTime){
        digitalWrite(dirPin, dir);
        if (dir == 0)
            dir = 1;
        else
            dir = 0;

        digitalWrite(stepPin, HIGH);
        digitalWrite(stepPin, LOW);
        delayMicroseconds(pause);
    }
}

/**
 *
 */
static void rest(int length)
```

```
{
    int endTime = millis() + length;
    while (millis() < endTime){
        delay(5);
    }
}

/**
 * song[note_num][note, octave, length]
 */
static void playSong(const int song[][3], const int tempo)
{
    // Convert tempo in BPM to millisecs
    int noteLen = 60000 / tempo;

    for (int i = 0; song[i][0] != -1; i++){
        int length = song[i][2] * noteLen;
        if (song[i][0] == Zz){
            rest(length);
        }
        else {
            playNote(song[i][0], song[i][1], (7 * length) / 8);
            rest(length / 8);
        }
    }
}

/**
 *
 */
int main()
{
    if (init() != 0){
        printf("init failed - Exiting\n");
        return 1;
    }

    playSong(song2, song2_tempo);

    return 0;
}
```