Communication between two Arduinos using the I2C bus.

### Tutorials:

<https://www.arduino.cc/en/Tutorial/MasterWriter>

(My Current code is based on this tutorial)

<https://www.arduino.cc/en/Tutorial/MasterReader>

<https://sandervandevelde.wordpress.com/2015/10/13/using-i2c-to-connect-two-arduino-nanos/>

(I will probably do a demo based on this code and my current code at some point)

<http://www.berryjam.eu/2014/07/advanced-arduino-i2c-communication/>

(another good tutorial)

<http://dsscircuits.com/articles/arduino-i2c-slave-guide>

(Some of my code is roughly based on this tutorial - Namely the idea of using a HEX code)

## The Master Device.

For this we are going to use D1 Mini with a OLED screen and 5 buttons. We are going to communicate with a I2C slave device with the ID of 0x08.

For now this a one way communication, and all we are going to do is have the slave play a

WAV file that is related to the button we push.

The code is highly based on:

<https://www.arduino.cc/en/Tutorial/MasterWriter>

The base from the tutorial is pretty easy - every half a second it sends a “message with counter” to the slave device. For the slave device to display.

We do this with

Wire.beginTransmission(8); //start a transmission to device 8

Wire.write(“x is “); //sends 5 bytes

Wire.write(x); // send one byte

Wire.endTransmission(); //stop

The counter is increased and the whole process is done over again

I wanted to be able to have the slave device do more than just display one thing. So for my code I added a few buttons, each button has a HEX code associated with it.

//I2C 0x00 status (Not used.)

#define BPIN D4 //I2C 0x01

#define APIN D3 //I2C 0x02

#define BUTTON1 D5 //I2C 0x03

#define BUTTON2 D6 //I2C 0x04

#define BUTTON3 D7 //I2C 0x05

I also define my REMOTE device address

#define REMOTE 0x08 //I2C address of slave device

You will notice that I switched to using HEX numbers - there really is no reason other than it looks cool! :-) And in a larger context a little easier to send large numbers this way.

The D1 Mini supports multiple interrupts so I set the buttons up on a interrupt pin.

Each time a button is pushed it will now send it’s HEX code to the slave.

For example: (This is the code for sending Button B the other buttons are the same with a different HEX number)

void SendoverI2C(int reg) {

Wire.beginTransmission(REMOTE); //transmit to REMOTE device

Wire.write(reg);

Wire.endTransmission();

}

void displayB() {

//I2C Register 0x01

display.clearDisplay();

display.setTextSize(5);

display.setTextColor(WHITE);

display.setCursor(10,10);

display.println("B");

SendoverI2C(0x01);

display.display();

}

I set up a function for sending different numbers to the slave (SendoverI2C) this saves on some code. The D1 Mini also displays the button pushed.

One of the keys to having the slave do multiple things is to have a plan on what you wish it to do. Make a plan and write your plan out so you can refer back to it when needed.

## The Slave Device.

For this Demo our Slave device is a MP3 player shield on top of a Arduino MEGA, the player plays a WAV file for each HEX code it gets.

The code is still based on the Arduino.cc Tutorial:

<https://www.arduino.cc/en/Tutorial/MasterWriter>

Again the base code in the tutorial is fairly easy to understand and use.

In the tutorial, it gets the message and counter and displays it in the serial monitor.

But in this case we have to tell Wire that the device has a number (or is a slave device)

We do that with

Wire.begin(8); //join I2C bus with address #8

We also have to setup a “software” interrupt also called a register event.

This is where the code is executed when the master sends a transmission.

Wire.onReceive(receiveEvent);

Next we need to receive the event:

void receiveEvent(int howMany) {

while (1 < Wire.available()) { *// loop through all but the last*

char c = Wire.read(); *// receive byte as a character*

Serial.print(c); *// print the character*

}

int x = Wire.read(); *// receive byte as an integer*

Serial.println(x); *// print the integer*

}

My code is very similar - There is of course some libraries and code for the playing of the WAV file, and that really isn’t important here.

My receiveEvent is pretty much the same as above with an added SWITCH/CASE

Each HEX number sent will cause the slave device to do something a little different.

void receiveEvent(int howMany) {

while (1 < Wire.available()) { // loop through all but the last

char c = Wire.read(); // receive byte as a character

//Serial.print(c);

}

int x = Wire.read();

//Serial.print(x);

/\*I2C registers

\* 0x00 Status

\* 0x01 B button

\* 0x02 A button

\* 0x03 Button 1

\* 0x04 Button 2

\* 0x05 Button 3

\*/

switch(x) {

case 0x01:

playFile("buttonb.wav");

break;

case 0x02:

playFile("buttona.wav");

break;

case 0x03:

playFile("one.wav");

break;

case 0x04:

playFile("two.wav");

break;

case 0x05:

playFile("three.wav");

break;

default:

break;

}

}

I tried a couple of different ways to receive the data, this is what seems to work for me.

You can see that each HEX number received will play a different file.

For these examples there is no feedback from the slave device. Next week I’d like to expand on this an have a sensor hooked to the slave and give feedback.