IAT 884

Interfacing with the Computer

IAT 884 Workshop Week 7

Interfacing with the Computer

1. Pull-Up Resistors Revisited (And Simplified)
2. Sending messages from the Arduino Board
3. Receiving messages using the Arduino Board
4. Receiving messages in Processing
5. Sending and Receiving Complex Messages
6. Converting received data
7. Using the ~~Minim~~ Sound Library
8. Workshop Assignment

Pull-Up Resistors Revisited

The Arduino's Built in Resistors

The Arduino has built in 20k Pull-Up Resistors on all digital pins. These can be activated by using the following two commands:

```
pinMode(pin, INPUT); // set pin to input
digitalWrite(pin, HIGH); // turn on pullup resistors
```

This simplifies your life tremendously. You can now connect a switch or pushbutton using only two wires.

The first wire connects one terminal on the switch to ground, and the other connects the second terminal to a digital input pin.

The connection to the Vcc (+) is done internally for you.

Serial Communication

Sending Messages: Arduino

```
void setup()
{
  Serial.begin(9600);          // ready serial comunication
}

void loop()
{
  Serial.print("Hello, ");     // prints a string
  Serial.println("world!");    // prints a string + newline
}
```

The message sent looks like this (H is the first character sent, so it's at the front of the message)

```
[10]-d-l-r-o-w- -,-o-l-l-e-H
```

When the message is received on the other end, it's output like this (if output as a string)

```
Hello, world!
```

Serial Communication

Sending Messages: Arduino

When you want to send the integer value 154 you write the command Serial.print(154);

The Arduino then sends every single digit as a separate information package: '1', '5', '4'.

Each package corresponds to a specific ASCII character that has a value between 0 -255 (1 Byte).

In this case:

'1' = 49
'5' = 53
'4' = 52

Serial Communication

Sending Messages: Arduino

```
int analogPin = 0;
int analogValue = 0;                // integer to print

void setup() {
  // open serial communications at 9600 bps
  Serial.begin(9600);
```

See https://www.arduino.cc/en/Serial/Print for updated version.

```
  Serial.print('\t');                // print a tab
  Serial.print(analogValue, DEC);    // print the ASCII encoded decimal analogValue
  Serial.print('\t');                // print a tab
  Serial.print(analogValue, HEX);    // print the ASCII encoded hexadecimal analogValue
  Serial.print('\t');                // print a tab
  Serial.print(analogValue, OCT);    // print the ASCII encoded octal analogValue
  Serial.println();                  // print a linefeed and carriage return

  delay(10);
}
```

- *Source: http://itp.nyu.edu/physcomp/Labs/Serial*

Serial Communication

Sending Messages: Arduino

Output from code:

BYTE	BIN	DEC	HEX	OCT
â	11100010	226	E2	342
á	11100001	225	E1	341
à	11100000	224	E0	340
ß	11011111	223	DF	337

•*Source: http://itp.nyu.edu/physcomp/Labs/Serial*

* Note: if no second argument is supplied to *Serial.print()* command the systems will default to DEC

Serial Communication

Receiving Messages: Arduino

```
int incomingByte = 0;   // for incoming serial data

void setup() {
        Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
}

void loop() {

        // send data only when you receive data:
        if (Serial.available() > 0) {
                // read the incoming byte:
                incomingByte = Serial.read();

                // say what you got:
                Serial.print("I received: ");
                Serial.println(incomingByte, DEC);
        }
}
```

- *Source:* https://www.arduino.cc/en/Serial/Read

Serial Communication

Receiving Messages: Processing

There are several functions that you can use to receive serial data in Processing:

read()
Returns a number between 0 and 255 for the next byte that's waiting in the buffer.

readChar()
Returns the next byte in the buffer as a char.

readString()
Returns all the data from the buffer as a String. This method assumes the incoming characters are ASCII.

Note that in this context a character value can be assigned to an integer variable (e.g. 'A' = 65), but that an integer value **cannot be assigned to a character variable.*

Serial Communication

Receiving Messages: Processing

```
// Example by Tom Igoe

import processing.serial.*;

// The serial port:
Serial myPort;

void setup() {
   // List all the available serial ports:
   printArray(Serial.list());
   // Open the port you are using at the rate you want:
   myPort = new Serial(this, Serial.list()[0], 9600);
}

void draw() {
   while (myPort.available() > 0) {
      int inByte = myPort.read();
      println(inByte);
   }
}
```

REMEMBER: The Arduino Serial Monitor & Processing will interfere with each other…

- *Source: https://processing.org/reference/libraries/serial/Serial_available_.html*

Serial Communication

Receiving Messages: Processing

If you are sending data that is between 0-255 you can send it from Arduino using:

```
Serial.println(156, BYTE);
```

and receive the value in processing using:

```
int inByte = myPort.read();
```

But this method will fail if you need to send a number over 255 for which there are no ASCII equivalents.

Serial Communication

Sending Messages: Processing

```
import processing.serial.*;

// The serial port:
Serial myPort;

// List all the available serial ports:
println(Serial.list());

// Open whatever port is the one you're using.
// WINDOWS probably:
myPort = new Serial(this, Serial.list()[0], 9600);
// MAC probably:
// myPort = new Serial(this, Serial.list()[1], 9600);

// Send a capital A out the serial port
myPort.write(65);
```

Serial Communication

Sending and Receiving Complex Serial Messages

Terminating Character:
Indicates when one message has ended and another has begun.

A good terminating character is the New Line Character (ASCII 10).

The following Arduino code sends a numeric message followed by a New Line character :

```
Serial.print(345943, DEC);
Serial.print(10, BYTE);
```

Serial Communication

Sending and Receiving Complex Serial Messages

```
void draw() {
  // See if there is data waiting to be received. in that case, trigger serial event:
  while(port.available() > 0){
    handleSerial(port.read());
  }
}

void handleSerial(int serial){
  // if number received is not a line feed (ASCII code 10)
  if(serial!=10) {
    // add partial data to buffer string (cast to character so it can be appended to a string)
    buf += char(serial);
  }
  else {  // we received terminating character!
    println(buf);
    // clear buffer
    buf="";
  }
}
```

Serial Communication

Converting Data: Processing

Once you have received data it is stored in a string variable. Often you will need to convert the data into a new type in order to do some kind of computation with it.

Example:

Incoming data stored in String variable buf = "1.234"

Conversion Options:

```
int intValue = int(buf);          //Casting data to an integer
```

intValue now stores the value 1

```
float floatValue = float(buf);    //Casting data to a float value
```

floatValue now stores the value 1.234

Serial Communication

The Minim Sound Library

```
import ddf.minim.*;

Minim minim;
AudioSample kick;

void setup()
{
```

We are just using the SOUND library.

See: https://processing.org/reference/libraries/sound/SoundFile.html

```
void keyPressed()
{
  if ( key == 'k' ) kick.trigger();
}
```

Serial Communication

The Minim Sound Library

Changing the audio sample

1. Place the new MP3 file in the data directory of the sketch
2. Change the name of the file “BD.mp3” to the name of your new sample.

```
void setup()
{
  size(512, 200, P2D);
  // always start Minim before you do anything with it
  minim = new Minim(this);
  kick = minim.loadSample("BD.mp3", 2048);
}
```

Serial Communication

The Minim Sound Library

Using the trigger() command

An example of this can be found in your processing examples folder under Libraries-->Minim(Sound)-->LoadSample

```
void keyPressed()
{
  if ( key == 'k' ) kick.play(); //sound library
}
```

There are many other options for playing samples using the minim library included in the reference. You are welcome to explore these methods as well.

http://code.compartmental.net/minim/javadoc/

Serial Communication

In Class Exercises

The goal of this workshop is to design a circuit that uses a push button switch to trigger a sound sample. The following steps break the task up into separate parts to make it easier to program the Arduino and build the circuit. If you want to skip all the steps and build everything in one go, that is fine as well.

A. Using serial communication, send a specific “trigger” number from the Arduino board to Processing. When the trigger number is received have Processing respond by displaying the number using the “println” command.

B. Modify your code from A to do the following: Instead of printing out the trigger number, have Processing play an audio sample using the Minim library.

C. Modify your code from B to do the following: Add a circuit that uses a pushbutton to play the audio sample. You should modify your code so that the Arduino board sends the trigger number when the button is pressed. (May require using “edge detection.”
See http://sproutlab.com/arduino/tutorials/3-serial-communication/ for help with this.)

OPTIONAL:
Modify your code in part C do the following: When processing receives the character number have it return a different number back to the Arduino. When the Arduino board receives this number it should light up the LED on pin 13 for 500 milliseconds.

Serial Communication

Edge Detection

The following code will count how many times a button is pressed.

```
int switchPin = 2;
int switchCounterVar = 0;
int switchStateVar = 0;
int lastSwitchStateVar = 0;

void setup() {
  pinMode(switchPin, INPUT);
}

void loop() {
  // read the switch
  switchStateVar = digitalRead(switchPin);
  // compare the switch to its previous state
  if (switchStateVar != lastSwitchStateVar) {
    // if the state has changed, increment the counter
    if (switchStateVar == HIGH) {
      switchCounterVar++;
    }
    // save the current state as the last state,
    //for next time through the loop
    lastSwitchStateVar = switchStateVar;
  }
}
```

Source: http://www.tigoe.net/pcomp/code/category/code/arduinowiring/43

Serial Communication Reference

Serial Communication Code

Arduino Commands https://www.arduino.cc/en/Reference/Serial

```
Serial.begin(9600);                          //initiate serial communication at 9600 baud

if (Serial.available() > 0) {                // checks to see if there is serial data waiting
        incomingByte = Serial.read();        // read the incoming byte to a variable
}

   Serial.print(100, DEC);                   //send the number 100 to the serial channel
   Serial.print(10, BYTE);                   // send the byte 100 to the serial channel
```

Processing commands https://processing.org/reference/libraries/serial/Serial.html

```
import processing.serial.*;                  //serial library

println(Serial.list());                      //List all the available serial ports:

String portname = "COM4";                    // variable to store the name of the com port

Serial port;                                 //Create a new serial port called port

port = new Serial(this, Serial.list()[1], 9600);   //opens serial communication port

while(port.available() > 0){                 //See if there is data waiting to be received.
     String buf = port.read();               //Read serial data into the string  variable buf
}

port.write(buf);                             //Send data stored in buf to the serial channel
port.write(100)                              //send the number 100 to the serial channel
```