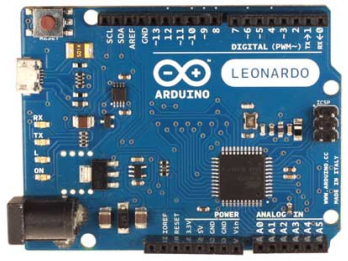
Digi-makers Arduino Tutorial

Introduction

Welcome! This tutorial will teach you how to use the wonderful Arduino modeling platform to create all sorts of electronic contraptions and games. This tutorial is aimed at those attending the Digi-makers@bristol events, however if you have the kit needed then the tutorial will be just as useful. Teachers are more than welcome to use it as materials in class as well.

First things first, **you will need:**



Arduino Leonardo Board

1. A Laptop / PC of your own with the arduino software installed. We will help you with this at the event.
2. An Arduino Board! We will be using the Leonardo model, but an older Uno board will work fine too. These will be provided at the event, along with the cables needed to connect them to your computer.
3. An arduino compatible LCD screen module. Again, we will provide these at the event. However if you are doing this at home the module we are using can be found here.
4. Any other components you can easily get your hands on, the more the merrier! You will need wires to connect things up and a breadboard to attach everything to, but we will provide these at the event. A breadboard is a special sort of circuit board that you can re-arange the components on without needing solder, very good for hobby work!

Have a look below for what we are going to make!

Pushing Buttons

First off, we are going to connect up some buttons and make some lights flash. Copy the code below into your Arduino IDE first. Then, use the diagram below to build the circuit. Don't be affraid to ask about the circuit. Better to get it right than blow something up! Remember, you don't need to copy the grey bits or anything after a "//". Those are just comments to help you understand the code more easily.

```
// Here we define which pins are which
#define BUTTON_PIN 5
#define LED_PIN 13

/*
  This function runs once when you first turn the Arduino on.
```

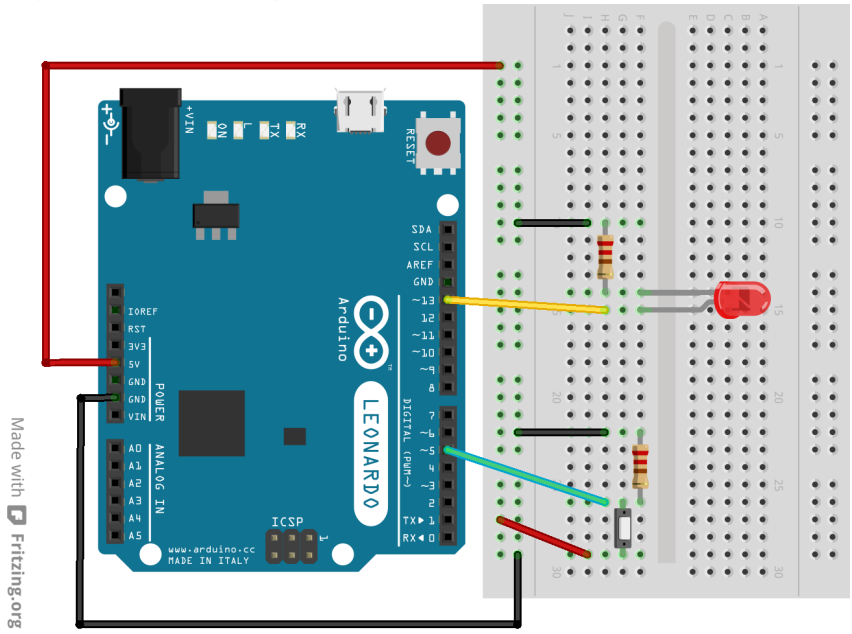
```
  It sets up which pins we are going to use for inputs and outputs.
 */
void setup()
{
  pinMode(BUTTON_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
}

/*
  After the setup function finishes, this one loops round and round
  until we turn the Arduino off again.
 */
void loop()
{
  // Check if the buttons are pressed. This is how you check if any
  // pin is turned on.
  int button1 = digitalRead(BUTTON_PIN);

  // if it is pressed, turn on light number 1
  if(button1 == HIGH)
  {
    // This is how we turn a pin on. We call turning it on "setting it high"
    digitalWrite(LED_PIN, HIGH);
  }
  else
  {
    // Again, this is how we turn a pin off. Called "setting it low"
    digitalWrite(LED_PIN, LOW);
  }
}
```

Now that you have the code all written, it is time to build the circuit. Use the diagram below to build it, and the photo for comparison when you finish. Ask for help if you aren't sure!

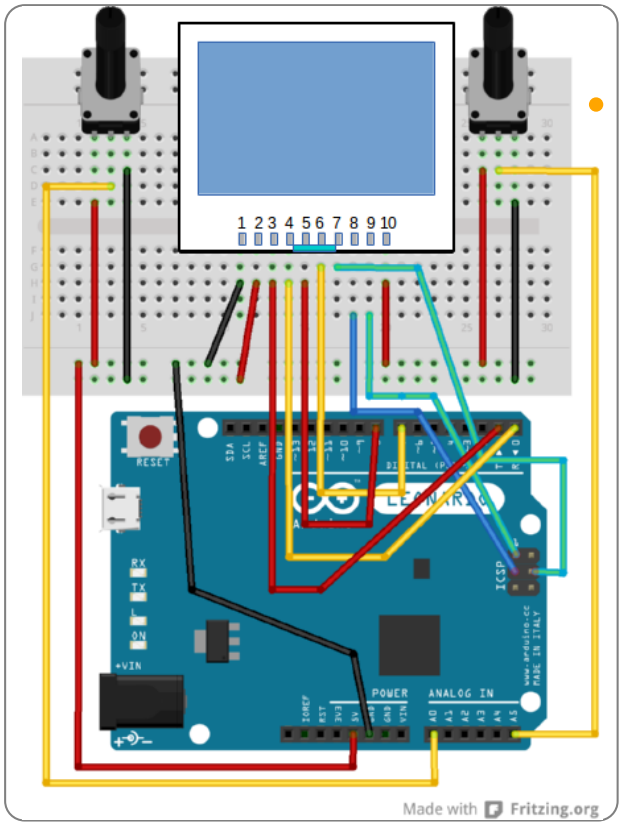
Also, remember to put the longer pin of the LED the right way round!



Once you have the code ready, save it somewhere and program your arduino! Make sure it is plugged in, then click on the "Upload" button and observe your handwork!

Using the Screen

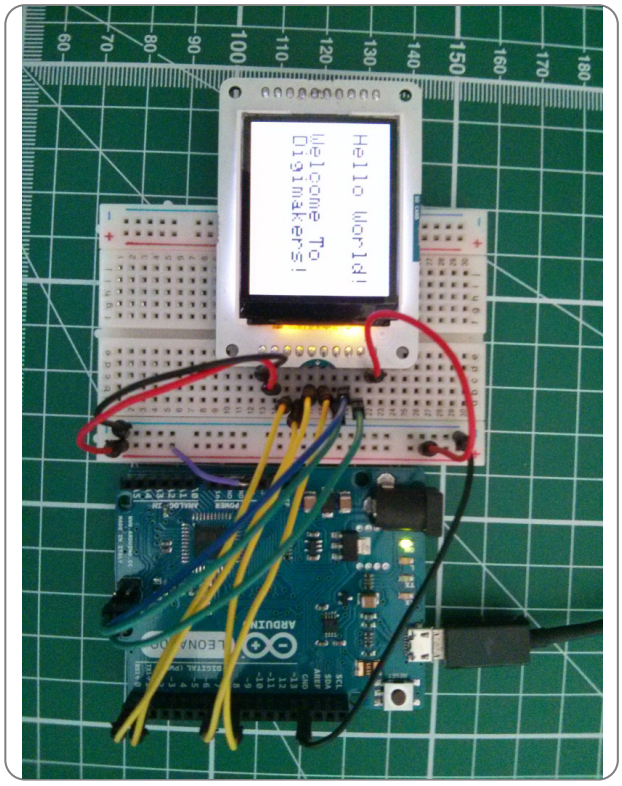
Right. Now the really fun part! What you do here is mostly up to you, all I am going to do is give you a few examples of what you can do with the screen. There will be prizes for good drawings, so get creative!



..

- **Getting started**

Whever you



program the screen, always start with this template code. You will need it every time you use the screen, so copy and paste it if you like.

```
// These two files contain everything the arduino needs to talk to the screen.
#include <SPI.h>
#include <TFT.h>

// create an instance of the library
TFT screen = TFT(7, 0, 1);

void setup() {
  // initialize the screen
  screen.begin();
  // In this function we put everything that we only want to draw once.
  // For example, make the background white:
  screen.background(255,255,255);
  // Add more stuff down here....
}

void loop() {
  // Set the text color, in this case black.
  screen.stroke(0,0,0);
  // Set how big the text should be. 1 = 10 pixels, 2 = 20 pixels etc...
  screen.setTextSize(2);
  // Write some text at position (X, Y)
  screen.text("Hello World!", 10, 20);
  // Add more stuff down here...
}
```

- ## Drawing Shapes

  The most basic thing you can do is draw shapes like rectangles, circles and lines. See if you can draw a little stick man or something. Try putting some of these in the setup function for now.

```
// Set the text and line color, in this case black.
screen.stroke(0,0,0);
// Draw a single dot on the screen
screen.point(screen.width()/2, screen.height()/2);
// Draw a line on the screen. (X start, Y start, X Stop, Y Stop)
screen.line(10, 100, 100, 10);
// Draw a rectangle on the screen (X start, Y start, width, height)
screen.rect(20, 20, 40, 60);
// Draw a circle on the screen (X, Y, size)
screen.circle(80, 80, 40);
```

- ## Writing Text

  This is always useful if you want to make a score counter....

```
// Set how big the text should be. 1 = 10 pixels, 2 = 20 pixels etc...
screen.setTextSize(1);
// Write some text at position (X, Y)
screen.text("Hello World!", 10, 80);
```

- ## Changing colors

  Because black and white is never enough, can you make the sky blue and the grass green around your stick man?

```
// Set the background color. Do this first or you will cover everything up!
screen.background(0,0,255);
// Set the colour of lines, and borders. Call this before line, point, circle or rect.
screen.stroke(255, 50, 100);
// And call this to remove the outlines and borders of the next shape you draw.
screen.noStroke();
// This sets the fill colour of shapes. Call this before text, circle or rect.
screen.fill(100, 100, 200);
// Can you guess what this does?
screen.noFill();
```

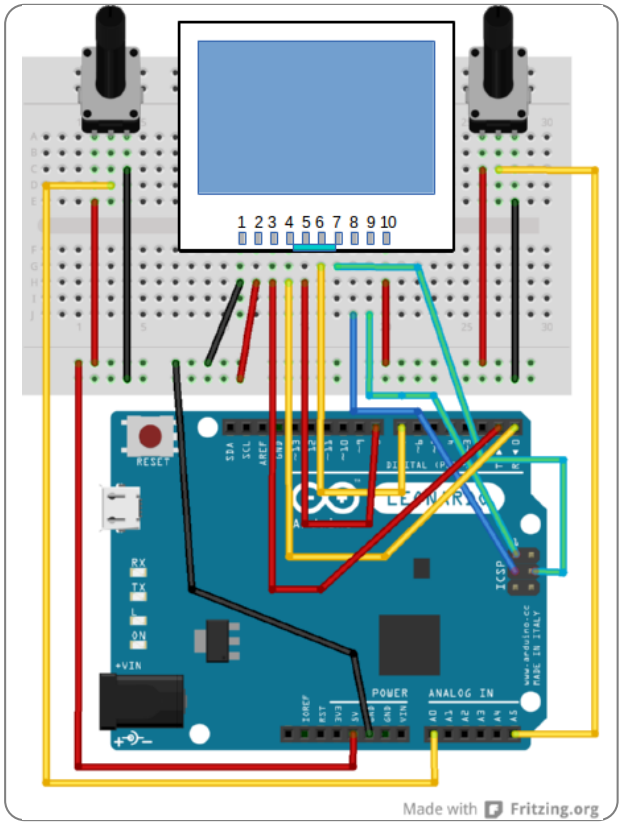
- ## Moving Things

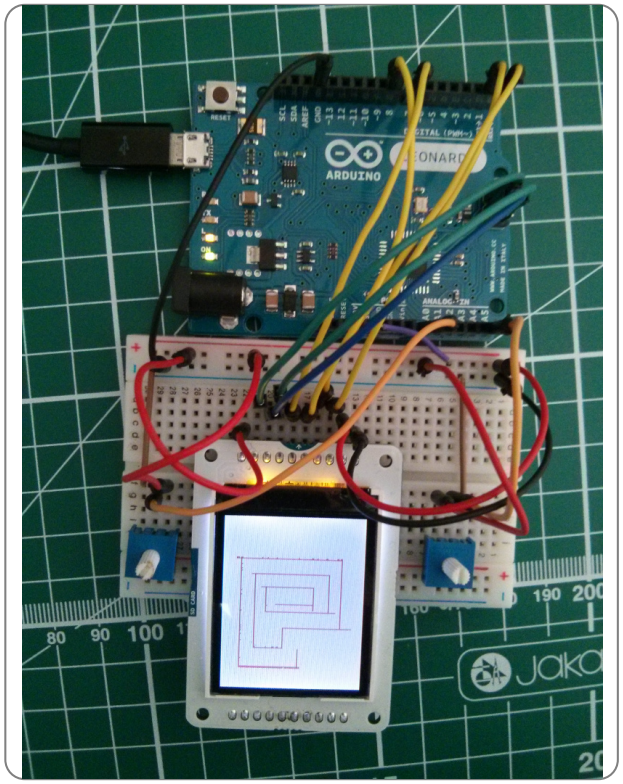
  Can you change something on your picture depending on if your button is pressed? Remember to put this code in your loop function, you might want to create a new file with the template for this bit as well. Here is a clue to get you going....

```
// This code moves a circle across the screen. Can you figure out how?
int x = 0;
int y = 0;
for(y=0; y<screen.height(); y++)
{
  screen.circle(x, y, 10);
  x += 1;
  delay(100);
}
// Can you make it change size as well? Can you make other shapes move?
```

Etch-a-sketch!

Now, lets get building! The first thing you will need to do is to build the circuit for your shiney new digital etch-a-sketch. Follow the diagram below and look at the photo for a reference of what the finished one should look like.





Okay, now we need some code! Your starting point is the template code for using the screen from the previous section. So get started by copying that in. The full example is shown below. See if you can work out what each piece of the code does. Brownie points for being able to explain it!

```
// These two files contain everything the arduino needs to talk to the screen.
#include <SPI.h>
#include <TFT.h>

// These are the pins each of the knobs connect to
#define Horizontal_Input A5
#define Vertical_Input A0

// create an instance of the screen
TFT screen = TFT(7, 0, 1);

// here is where we will store the position of the cursor
int h_pos = 0;
int v_pos = 0;

void setup() {
  screen.begin();
  screen.background(255,255,255);
}

void loop() {
  // First, lets update where we think the cursor is
  h_pos = analogRead(Horizontal_Input);
```

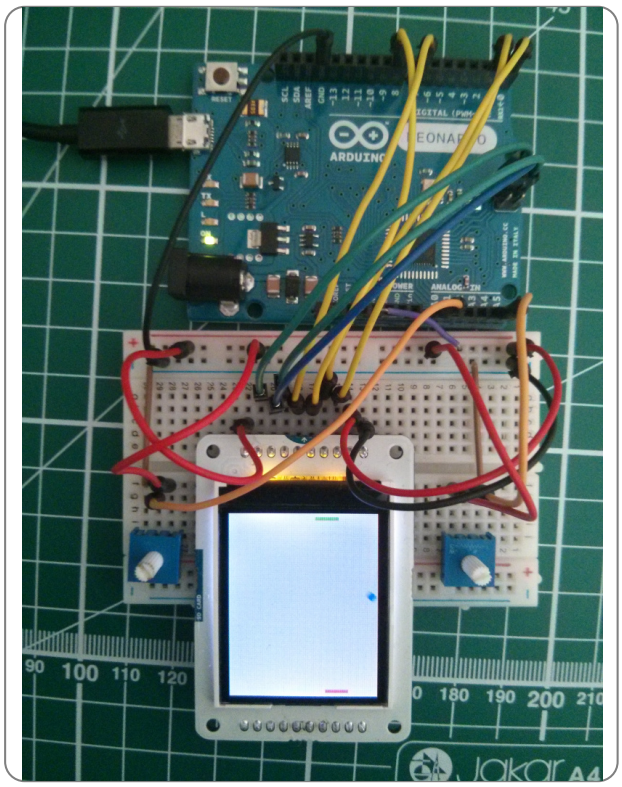
```
  v_pos = analogRead(Vertical_Input);
  // What do you think these lines do? Have a look below
  // the code to find out.
  h_pos = map(h_pos, 0, 1023, 0, screen.width());
  v_pos = map(v_pos, 0, 1023, 0, screen.height());
  // Now lets draw the dot where the cursor is:
  // first set the color:
  screen.stroke(255,0,0);
  // Now draw the point:
  screen.point(h_pos,v_pos);
}
```

If you manage to get it all working (which ofcourse you will) then have a go at customising it a little. Can you change the colours? Can you make it draw red in the right half of the screen and blue in the left half?

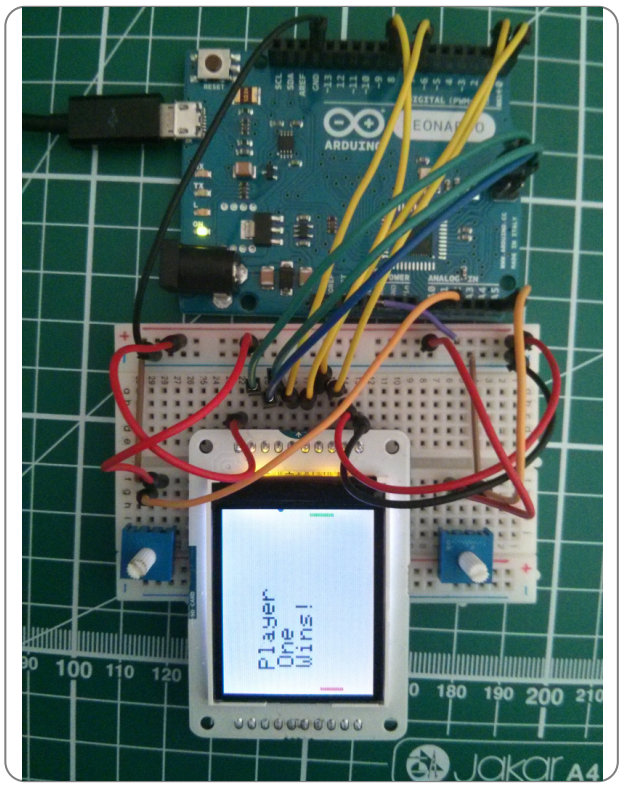
Go Crazy!

Playing Pong!

How about a game? We are going to build a replica of the first ever computer arcade game, pong! This is a bit more complicated, but I reckon you can manage it. Again, first we need to build the circuit, luckily. It is the same as the one for the etch-a-sketch. How convinient!



There is a bit more code for pong,



however if you understood the etch-a-sketch then this should be no problem. Just copy it out first to get it working, then see if you can go through it line by line to try and understand it. Please do ask if you want to try and explain it to us, or if you would like us to explain it to you!

```
// These two files contain everything the arduino needs to talk to the screen.
#include <SPI.h>
#include <TFT.h>

// These are the pins each of the knobs connect to
#define Horizontal_Input A5
#define Vertical_Input A0

// create an instance of the screen
TFT screen = TFT(7, 0, 1);

// How big the paddels will be:
int bat_size = 20;
// here is where we will store the position of the players
int p1_pos = 0;
int p2_pos = 0;
// Store their previous positions too. You'll see why in a second...
int old_p1_pos = 0;
int old_p2_pos = 0;
// Store the position of the ball
int ball_x = screen.height()/4;
int ball_y = screen.width()/2;
int old_ball_x = screen.height()/2;
int old_ball_y = screen.width()/2;

int ball_speed_x = 1;
int ball_speed_y = 1;


void setup() {
 screen.begin();
 screen.background(255,255,255);
 screen.setTextSize(2);
}

void loop() {
 // First, lets update and draw where the player paddles are
 // Can you remember what these lines did from the last example?
 p1_pos = analogRead(Horizontal_Input);
 p2_pos = analogRead(Vertical_Input);
 // Why do we subtract 20 from the screen height?
 p1_pos = map(p1_pos, 0, 1023, 0, screen.height()-20);
 p2_pos = map(p2_pos, 0, 1023, 0, screen.height()-20);

 // Now lets draw the paddles.
 if(p1_pos != old_p1_pos || p2_pos != old_p2_pos)
 {
  screen.fill(255,255,255);
  screen.rect(3, old_p1_pos, 3, bat_size); // Player 1
  screen.rect(154, old_p2_pos, 3, bat_size); // Player 1
 }

 // Draw the player paddles
 screen.fill(255,0,0); // Player 1 paddle colour.
 screen.rect(3, p1_pos, 3, bat_size); // Player 1
 screen.fill(0,255,0); // Player 2 paddle colour.
 screen.rect(154, p2_pos, 3, bat_size); // Player 2

 // Draw the ball!
 screen.fill(255,255,255);
 screen.circle(old_ball_x,old_ball_y,3); // erase the old ball
 screen.fill(0,0,255);
 screen.circle(ball_x,ball_y,3); // draw the new one!
```

```
  // Now we need to deal with moving the ball and making sure that is
  // bounces off the edges properly!
  if(ball_y > screen.height() || ball_y < 0)
  {
    ball_speed_y = -ball_speed_y;
  }

  // check if the ball has gone off the end of the screen
  if(ball_x > screen.width())
  {
    screen.stroke(0,0,0);
    screen.text(" Player\n  One\n  Wins!",10,30);
    delay(10000);
  }
  else if (ball_x < 0)
  {
    screen.stroke(0,0,0);
    screen.text(" Player\n  Two\n  Wins!",10,30);
    delay(10000);
  }
  else
  {
    // Has it hit player one's bat?
    if(ball_x < 4 && ball_y < p1_pos + bat_size && ball_y > p1_pos)
    {
       ball_speed_x = -ball_speed_x;
    }
    // Has it hit player two's bat?
    else if(ball_x > screen.width()-6 && ball_y < p2_pos + bat_size && ball_y > p2_pos)
    {
       ball_speed_x = -ball_speed_x;
    }
  }

  // Update the old positions
  old_p1_pos = p1_pos;
  old_p2_pos = p2_pos;
  old_ball_x = ball_x;
  old_ball_y = ball_y;
  ball_x = ball_x + ball_speed_x;
  ball_y = ball_y + ball_speed_y;

  delay(10);
}
```

Extra Challenges: Can you make a score counter? Can you make each block a different color, say red and blue for each player? Change the shape of the ball?

Again, **Go Crazy!**

And Finally...

Thank you for taking part in this workshop and using this tutorial! If you would like some more information you can always get in touch with me via email: *benmarshall@gmx.co.uk*. I am also on LinkedIn.

If you are a teacher and would like to know how this workshop can be adapted for use in classrooms and schools, contact me at *outreach@cssbristol.co.uk*. I am always happy to work with teachers in creating lesson plans and helping with any and all technical aspects of the new computing curriculum. You can freely download a PDF version here

For more ideas, tutorials and materials, check out the Arduino website.

Written and Devised by Ben Marshall: benmarshall *at* gmx.co.uk
bmarshall.org.uk | LinkedIn