

Welcome! This workshop will show you how to build and program your own arcade games using the Arduino electronics platform.

1 - Getting Started

This step tells you the materials and software you will need to complete this workshop.

2 - Using the Screen

Here, we will build the circuit that connects the screen to the Arduino, and learn how to display basic shapes on the screen. We will also go over how to connect the Arduino to the computer and program it using the Arduino IDE.

3 - Inputs

All arcade games need controls! This section will show you how to connect buttons and dials to your Arduino and write code to interact with them. We can then start displaying things on the screen based on whether our buttons are being pressed!

4 - Let's Make Games!

Now we know everything we need to make some games! This section details a few ideas to get you started, and gives some example code for ping-pong. If you can complete this section, carry on and make the coolest game you can!

Arduino Games

Arduino Games

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build simple arcade games using the Arduino platform, an LCD screen module and other basic electronic components.

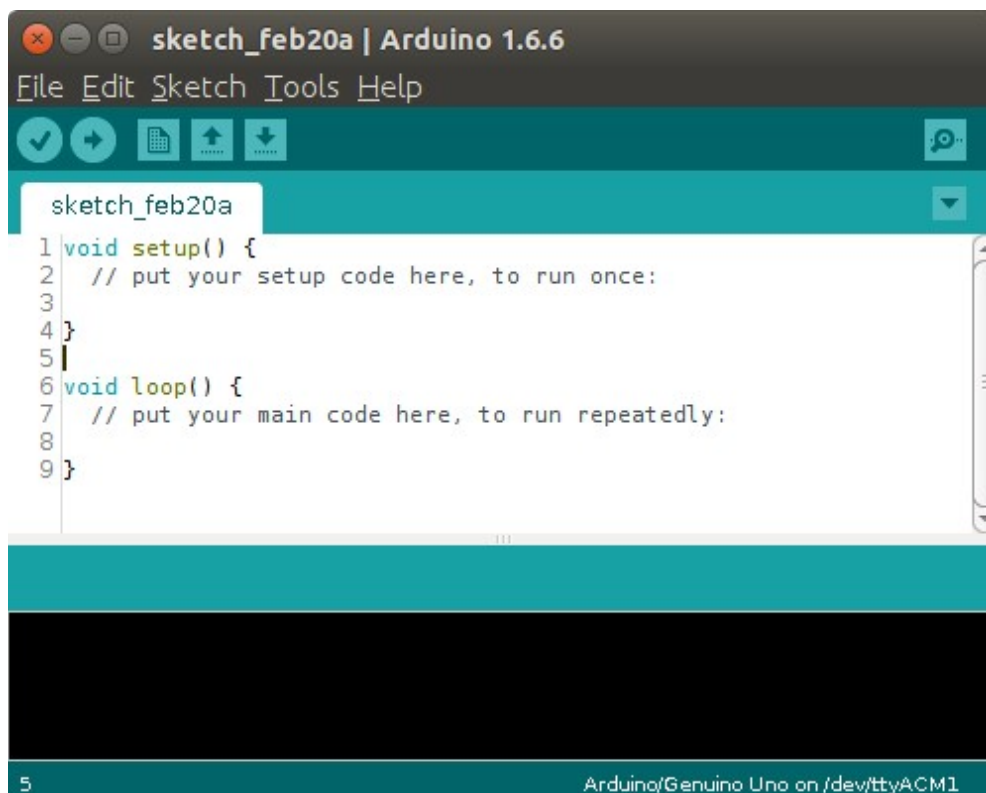
Getting Started

This page will tell you how to get the Arduino software installed on your computer, and run a basic program to check everything is working.

1. Downloading The Software

First, head over to [the Arduino website](#) and download the latest version of their software for your computer. They do versions for Windows, Max OS X and Linux.

Once you have downloaded and installed the Arduino software, launch it and you should see a window like the one shown in the image below.



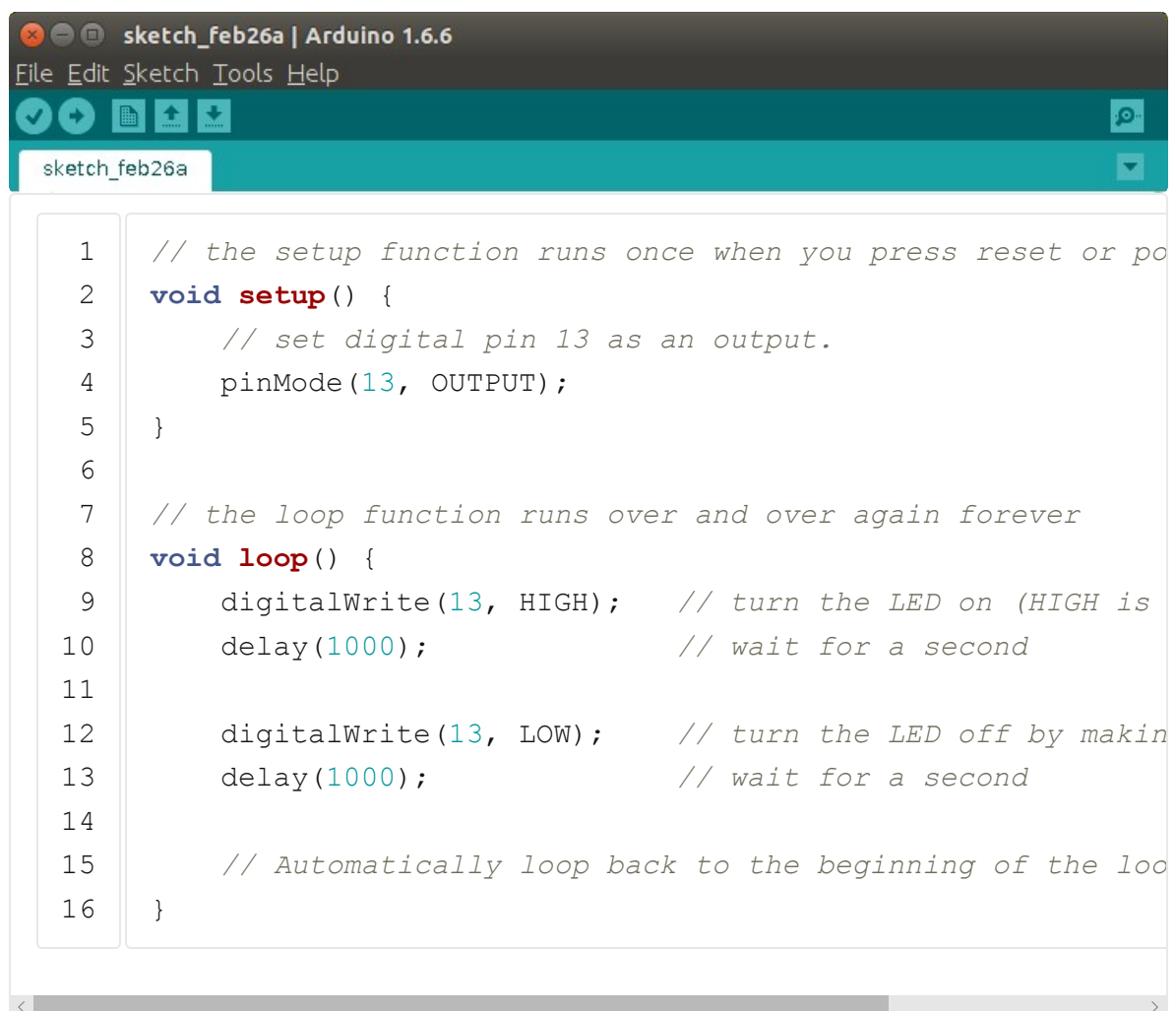
2. Connecting The Board

Next, we need to actually connect our Arduino Board to the computer. Use the USB cable provided, and plug it into the mini-USB port on the Arduino, and a USB port on your PC. You should see the little green LED on the board light up.

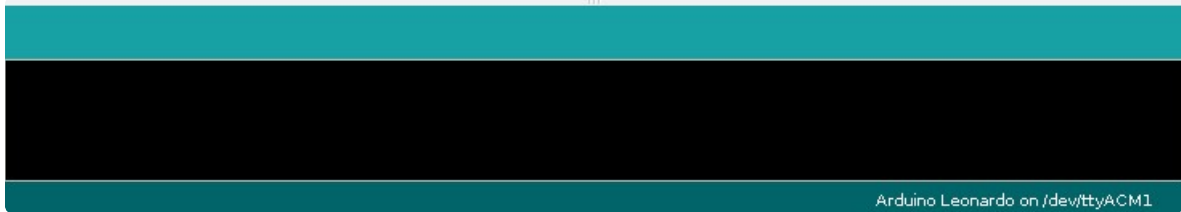
Now to set up the connection to the board. Using your Arduino software, click on the **Tools** menu icon, and under the *Board:* sub-menu, select the board you are using. For the Digimakers workshop, we are using the "*Arduino Leonardo*".

With the correct board selected, click again on the **Tools** menu, and this time click on the "*Port:*" sub-menu. What you will see will depend on your operating system. Windows users will see "COMX" where X is a number, while Linux users will usually see "/dev/ttyACMX". Select the first one in the list. You can try the others if the first one doesn't work in the next step.

Back on your Arduino IDE, goto the **File** menu and select *Examples -> 01. Basics -> Blink*. A new window will appear with some code already inserted. You can now click on the *Upload* button in the **Sketch** menu, or, for wizz-kids, press *Ctrl-U*.

A screenshot of the Arduino IDE 1.6.6 interface. The title bar reads "sketch_feb26a | Arduino 1.6.6". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for opening, saving, uploading, and downloading. The main text area shows the code for the "Blink" example, with line numbers 1 through 16 on the left. The code is as follows:

```
1 // the setup function runs once when you press reset or po
2 void setup() {
3     // set digital pin 13 as an output.
4     pinMode(13, OUTPUT);
5 }
6
7 // the loop function runs over and over again forever
8 void loop() {
9     digitalWrite(13, HIGH);    // turn the LED on (HIGH is
10    delay(1000);               // wait for a second
11
12    digitalWrite(13, LOW);     // turn the LED off by makin
13    delay(1000);               // wait for a second
14
15    // Automatically loop back to the beginning of the loo
16 }
```



Some text should fly past a the bottom of your Arduino IDE window, the last line of which will be "Upload Complete!". The yellow LED on your Arduino should now be flashing!

3. Next Steps

Now we know we can upload code to our Arduino, can you work out what our "Blink" program does, and how? We'll go over the code properly in the next section, but the more you can figure out now, the better.

In the next section, we will assemble the circuit to connect the screen to the Arduino.

[< Home](#) | [Using The Screen](#) >

Arduino Games

Arduino Games

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-
Outreach](#)

 [digi_makers](#)

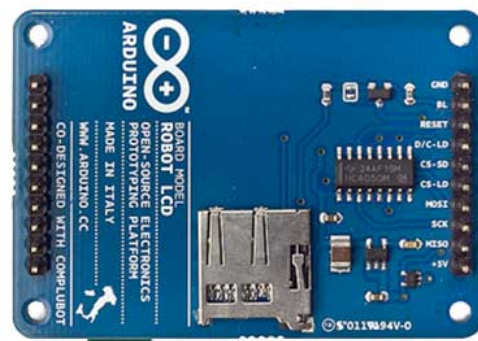
A tutorial written for the Digimakers project that teaches people how to build simple arcade games using the Arduino platform, an LCD screen module and other basic electronic components.

Using The LCD Screen

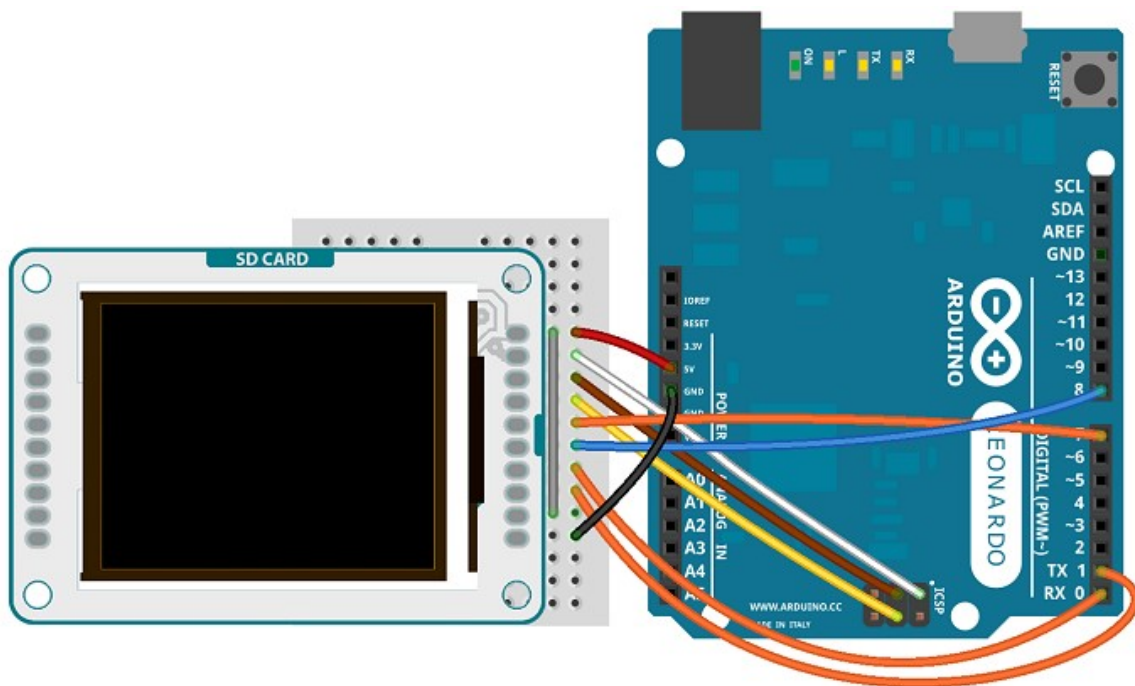
In this section, we will build the circuit that connects the LCD screen to the Arduino, and learn how to program the screen to display different shapes, colours and text.

Building The Circuit

We will be using the Arduino LCD module as our screen. You can find more information on the screen [here](#), and we will show you how to wire it up properly below.



The wiring diagram for the screen is shown below. It assumes you are using an Arduino Leonardo board. The breadboards we are using are slightly bigger than the one shown, but that's okay. As long as you get the right wires in the right place!



Made with  Fritzing.org

Checking The Circuit

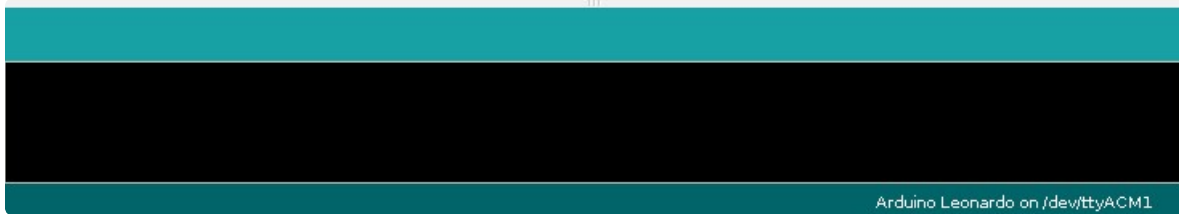
To check we have connected the screen properly, lets run a small "Hello World!" program. Open up your Arduino software, and use the code supplied below to check it works.

```
sketch_feb26a | Arduino 1.6.6
File Edit Sketch Tools Help
[Icons]
sketch_feb26a

1  #include <TFT.h>  // Arduino LCD library
2  #include <SPI.h>
3
4  // pins to control the screen
5  #define CS      7
6  #define DC      0
7  #define RST     1
8
9  // create an instance of the library
10 TFT screen = TFT(CS, DC, RST);
11
12 // Use this to count upwards forever.
13 int counter = 0;
14
15 // array of letters to print to the screen
```



```
16 char letters[10];
17
18 void setup() {
19
20 // Put this line at the beginning of every sketch that use
21 screen.begin();
22
23 // clear the screen with a white background
24 screen.background(255,255,255);
25 // Set the fill colour to white.
26 screen.fill(255,255,255);
27 // set the font colour to black
28 screen.stroke(0, 0, 0);
29 // set the font size
30 screen.setTextSize(2);
31 // ste the font size very large for the loop
32 screen.setTextSize(2);
33 }
34
35 void loop() {
36 // Write hello world to the screen.
37 screen.text("Hello World!", 1, 20);
38
39 // The screen is dumb and always writes over itself.
40 // Blank the part of the screen that will change.
41
42 screen.rect(1,40,100,40);
43
44 // Print out the counter number.
45 String count = String(counter);
46 count.toCharArray(letters,10);
47 screen.text(letters, 1,40);
48
49 // wait for a moment then loop back round.
50 delay(500);
51 counter = counter + 1;
52 }
```

Hit **Upload** again, and hopefully, you will see the words "Hello World!" appear on your screen.

Programming The Screen

Now we will learn to program the screen and actually draw interesting stuff on it.

The code below goes through a bunch of different examples, and shows you the different things the screen you can do. Complex things can be drawn from lots of simple things, and you can build them into a full game. Try copying the code below and seeing how it behaves when you run it. Once you think you know how each bit works, change things and see how it effects what gets displayed.

This is called **Hacking!**

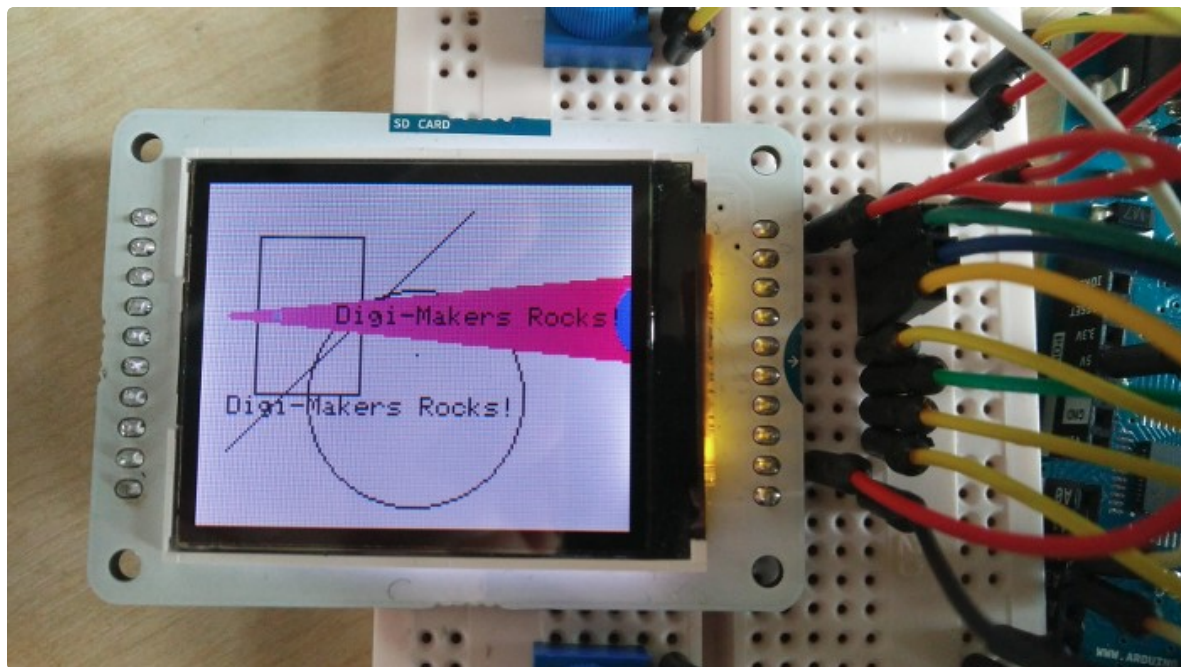
```
1  // These two files contain everything the arduino needs to
2  #include <SPI.h>
3  #include <TFT.h>
4
5  // create the screen object we can use to draw things with
6  TFT screen = TFT(7, 0, 1);
7
8  void setup() {
9    // initialize the screen
10   screen.begin();
11   // In this function we put everything that we only want
12   // For example, make the background white:
13   screen.background(255,255,255);
14   screen.stroke(0,0,0);
15   // Add more stuff down here....
16
17   // Draw a single dot on the screen
```

```

18     screen.point(screen.width()/2, screen.height()/2);
19     // Draw a line on the screen. (X start, Y start, X Stop,
20     screen.line(10, 100, 100, 10);
21     // Draw a rectangle on the screen (X start, Y start, X S
22     screen.rect(20, 20, 40, 60);
23     // Draw a circle on the screen (X, Y, size)
24     screen.circle(80, 80, 40);
25
26     // Text
27     // Set how big the text should be. 1 = 10 pixels, 2 = 20
28     screen.setTextSize(1);
29     // Write some text at position (X, Y)
30     screen.text("Digi-Makers Rocks!", 10, 80);
31
32     delay(1000);
33     // Colours
34     // Set the background color. Do this first or you will c
35     // screen.background(0,0,255);
36     // Set the colour of lines, and borders. Call this befor
37     screen.stroke(255, 50, 100);
38     // And call this to remove the outlines and borders of t
39     //screen.noStroke();
40     // This sets the fill colour of shapes. Call this before
41     screen.fill(100, 100, 200);
42     // Can you guess what this does?
43     //screen.noFill();
44 }
45
46 void loop() {
47     // In here is where we put everything we want to continu
48     // on the screen
49
50     // This code moves a circle across the screen. Can you f
51     int x = -10;
52     int y = 0;
53     for(y=0; y<screen.width()+20; y++)
54     {
55         screen.circle(x, 50, 10);
56         x += 1;
57         delay(20);
58     }
59     screen.stroke(0,0,0);
60     screen.text("Digi-Makers Rocks!", 50, 46);

```

```
61     screen.stroke(255, 50, 100);  
62     // Can you make it change size as well? Can you make oth  
63 }
```





Now you feel comfortable using the screen and the code to control it, lets move on to making our screen interactive. This will involve buttons, knobs, circuits and a little more code...

< Getting Started | Buttons and Dials >

Arduino Games

Arduino Games
Written By [Ben Marshall](#)
bm12656@my.bristol.ac.uk

 [MVSE-](#)
[Outreach](#)
 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build simple arcade games using the Arduino platform, an LCD screen

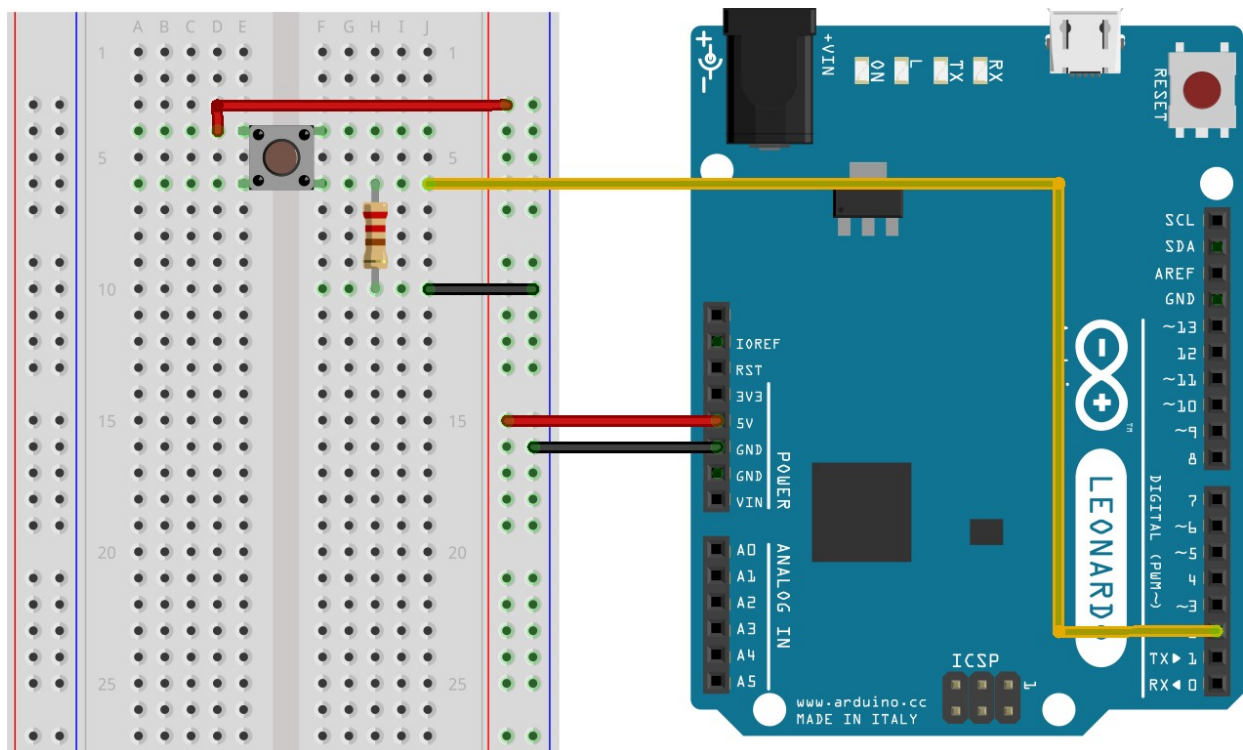
module and other basic electronic components.

Pushing Buttons

Now we know how to use the screen, we need to make it more interactive. This section will show you how to connect buttons and dials to your Arduino, and use code to *read* a value from them.

Adding A Button

Let's start simple. We can really easily connect a button to our arduino using only two wires and a little code. The diagram below shows how you can do this. You'll notice that the diagram doesn't include the screen. That's just to make things clearer, you'll have to work out how to fit the buttons around the screen so you can press them easily later.



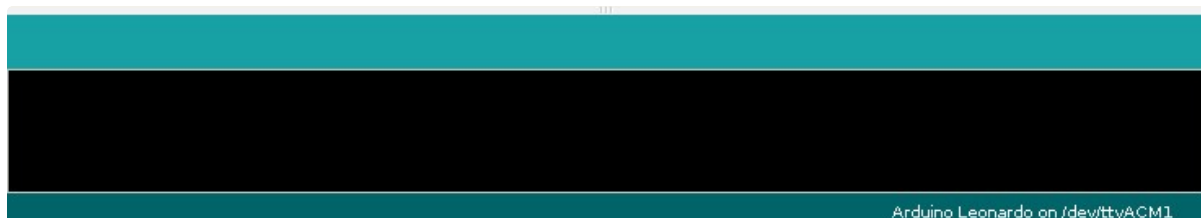
With the circuit all wired up, you can start coding. The code below can be used along side the code from your screen. First though, lets try doing something really simple like blinking a light when you press the button.



```

1  // These constants won't change. They're used here to
2  // set pin numbers:
3  const int buttonPin = 2;    // the number of the pushbutton pin
4  const int ledPin = 13;     // the number of the LED pin
5
6  // Store whether the button is on or off here.
7  int buttonState = 0;
8
9  void setup() {
10     // Initialize the LED pin as an output:
11     pinMode(ledPin, OUTPUT);
12     // Initialize the pushbutton pin as an input:
13     pinMode(buttonPin, INPUT);
14 }
15
16 void loop() {
17     // read the state of the pushbutton value:
18     buttonState = digitalRead(buttonPin);
19
20     // Check if the pushbutton is pressed.
21     // If it is, the buttonState is HIGH:
22     if (buttonState == HIGH) {
23         // Turn LED on:
24         digitalWrite(ledPin, HIGH);
25     } else {
26         // Turn LED off:
27         digitalWrite(ledPin, LOW);
28     }
29 }

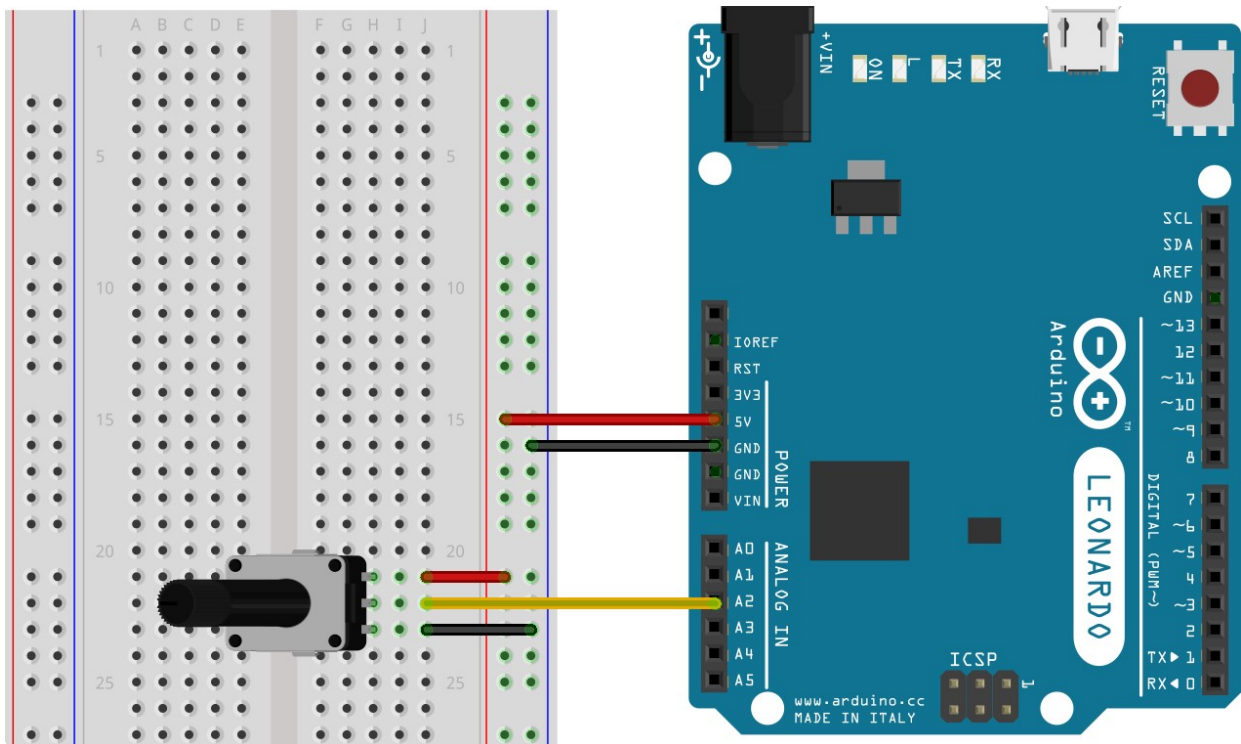
```



You should see that whenever you press the button, the light turns on! Not very exciting, but think about how we can use this code to build our games? Can you make something move on your screen based on button presses?

Adding A Dial

Buttons are great and all, but what about something twisty? Dials can be used move characters, change speed, steer and all sorts of things. They are nice and simple to wire up too. Again, the wiring diagram below only shows the dial, can you fit it around the screen too?



Now we have everything wired up, let's make something on our screen change based on our dial. When we read a value from the dial, we get a number between 0 and 1023. We can use this as a position, or even a colour. The code below will move a square up and down, and also change its colour, based on the position of the dial. Can you make the rectangle move differently if the button is pressed as well? What about changing the shape?

```

sketch_feb26a | Arduino 1.6.6
File Edit Sketch Tools Help

sketch_feb26a

1  #include <TFT.h>  // Arduino LCD library
2  #include <SPI.h>
3
4  // pins to control the screen
5  #define CS      7
6  #define DC      0
7  #define RST     1
8
9  // Pin for the dial
10 #define DIAL A2
11
12 // create an instance of the library
13 TFT TFTscreen = TFT(CS, DC, RST);
14
15 // Use this to store the dial value
16 int dial = 0;
17 // Store the position of our bat.
18 int pos = 0;
19

```



```

20 void setup() {
21
22 // Put this line at the beginning of every sketch that uses the GLCD:
23 TFTscreen.begin();
24
25 // clear the screen with a white background
26 TFTscreen.background(255,255,255);
27 // Set the fill colour to white.
28 TFTscreen.fill(255,255,255);
29 // set the font colour to black
30 TFTscreen.stroke(0, 0, 0);
31
32 }
33
34 void loop() {
35
36 // The screen is dumb and always writes over itself.
37 // Blank the part of the screen that will change.
38 TFTscreen.fill(255,255,255);
39 TFTscreen.stroke(255, 255, 255);
40 // Draw over the rectangle.
41 TFTscreen.rect(1,pos,10,20);
42
43 // Read the value of the dial.
44 dial = analogRead(DIAL);
45 pos = dial / 4;
46
47 // Set the fill colour to green.
48 TFTscreen.fill(0,255,0);
49 // set the line colour to red
50 TFTscreen.stroke(255, 0, 0);
51 // Draw the rectangle.
52 TFTscreen.rect(1,pos,10,20);
53
54 delay(10);
55
56 }

```

111

Arduino Leonardo on /dev/ttyACM1

Congratulations! Now you know everything you need in order to start making some awesome arcade games. The next section will show you how to build the classic *ping pong*, and give you some ideas about other games you could try.

< Using The Screen | Ping Pong >

Arduino Games

Arduino Games

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build simple arcade games using the Arduino platform, an LCD screen module and other basic electronic components.

Ping Pong

We have everything we need, we've seen the code, the circuit is built. Lets make a game! We'll start by breaking the game down into steps. This will make it easier to develop, and is called *divide and conquer*, a really common strategy for solving hard problems with code.

Divide and Conquer

So, how many steps are there in making ping pong? It can be supprisingly complicated if you don't think about it a little first! We need a score keeper, two bats, collision detection, input read code, and to display all this on the screen. All fast enough so it doesn't run at a snails pace! Harder than you thought huh? Never fear, we will build things up step by step. Below is a list of all the stages we will go through in building the game.

1. Keeping Score
2. Player Bats
3. Moving The Ball And Collisions
4. Winning!

Don't worry if you get stuck. The workshop helpers will be happy to lend a hand. You can also take a quick peek at the finished code [here](#) if you are really stuck.

Keeping Score

Pong is a two player game, so we need to have two scores, and to be able to read inputs from both players individually. Below is some example code with small bits missing, can you fill them in so we can read the position of the bats properly and display each players score?



```
1  #include <TFT.h> // Arduino LCD library
2  #include <SPI.h>
3
4  // pins to control the screen
5  #define CS 7
6  #define DC 0
7  #define RST 1
8
9  // Pins for the player paddle positions
```

```

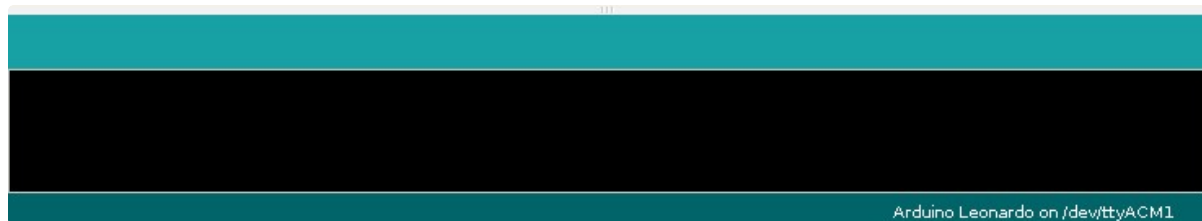
10 #define PIN_P1_BAT A2
11 #define PIN_P2_BAT A3
12
13 // create an instance of the library
14 TFT screen = TFT(CS, DC, RST);
15
16 // Player Scores
17 int score_p1;
18 int score_p2;
19 int position_p1;
20 int position_p2;
21
22 void drawPlayerScores()
23 {
24     // This function will draw a very simple "X - Y" string where
25     // X and Y are the scores for each player.
26
27     // Set the text color to black.
28     screen.stroke(0,0,0);
29
30     // We will print five letters, so make our array five long.
31     char toPrint[5];
32     String S = "";
33     S += String(score_p1); // Add the player 1 score
34     S += " - ";           // Separate the two scores
35     // .....             // Add the player 2 score
36     S.toCharArray(toPrint,5); // Convert the string to something we can
37     // .....             // Display the text!
38 }
39
40 void getPlayerPositions(){
41     // Read player 1 position
42     // What do you think the map function does?
43     position_p1 = map(analogRead(PIN_P1_BAT),0,1023-bat_h,0,screen_h);
44
45     // Read player 2 position
46     // ....
47 }
48
49 void setup() {
50
51     // Set up the screen.
52     screen.begin();
53     screen.background(255,255,255);
54     screen.fill(255,255,255);
55     screen.stroke(0, 0, 0);
56     screen.setTextSize(1);
57
58     // Set up the player scores and bat positions.
59     score_p1 = 0;
60     score_p2 = 0;

```

```

61     position_p1 = 0;
62     position_p2 = 0;
63 }
64
65 void loop() {
66
67     // Read the player positions.
68     getPlayerPositions();
69
70     // Draw the player scores.
71     drawPlayerScores();
72
73     delay(500);
74 }

```



Got it working? This code will form the skeleton of our game, so all of the other code will be **added** to what we have already written.

Player Bats

So, we can draw the player scores, and read where the bats *should* be. But we don't draw the bats! Let's add a function that uses the positions we read in the previous step to draw them in.



```

15
16  const int bat_w = 5;
17  const int bat_h = 10;
18
19  const int screen_w = 320;
20  const int screen_h = 240;
21
22  // Player Scores
23  int score_p1;
24  int score_p2;
25  int position_p1;
26  int position_p2;
27
28  void drawPlayerBats()
29  {
30      // Draw player 1's bat. First, clear the old area the bat took up.
31      screen.fill(255,255,255);
32      screen.stroke(255,255,255);
33      screen.rect(1,position_p1,bat_w,bat_h);
34
35      // Clear player 2's bat. How do we know the position of their bat.
36      // .....
37
38      // Read the player positions. This changes position_p1 and position_p2
39      getPlayerPositions();
40
41      // Now we can draw the new bats
42      // Draw player 1's bat first.
43      screen.fill(0,255,0); // Make player 1's bat green.
44      screen.rect(1,position_p1,bat_w,bat_h);
45
46      // Now draw player 2's bat. Remember, it needs to be on the other
47      // of the screen.
48      // .....
49  }
50
51  void drawPlayerScores()
52  {
53      // This function will draw a very simple "X - Y" string where
54      // X and Y are the scores for each player.
55
56      // Set the text color to black.
57      screen.stroke(0,0,0);
58
59      // We will print five letters, so make our array five long.
60      char toPrint[5];
61      String S = "";
62      S += String(score_p1); // Add the player 1 score
63      S += " - ";           // Separate the two scores
64      // .....           // Add the player 2 score
65      S.toCharArray(toPrint,5); // Convert the string to something we can

```

```

66         // .....          // Display the text!
67     }
68
69     void getPlayerPositions(){
70         // Read player 1 position
71         position_p1 = analogRead(PIN_P1_BAT) / 4;
72
73         // Read player 2 position
74         // ....
75     }
76
77     void setup() {
78
79         // Set up the screen.
80         screen.begin();
81         screen.background(255,255,255);
82         screen.fill(255,255,255);
83         screen.stroke(0, 0, 0);
84         screen.setTextSize(1);
85
86         // Set up the player scores and bat positions.
87         score_p1      = 0;
88         score_p2      = 0;
89         position_p1 = 0;
90         position_p2 = 0;
91     }
92
93     void loop() {
94
95         // Move the "getPlayerPositions()" function call into the
96         // new drawPlayerBats() function. Can you figure out why?
97
98         // Draw the player scores.
99         drawPlayerScores();
100
101         // Draw the bats using the positions we read a moment ago.
102         drawPlayerBats();
103
104         delay(500);
105     }

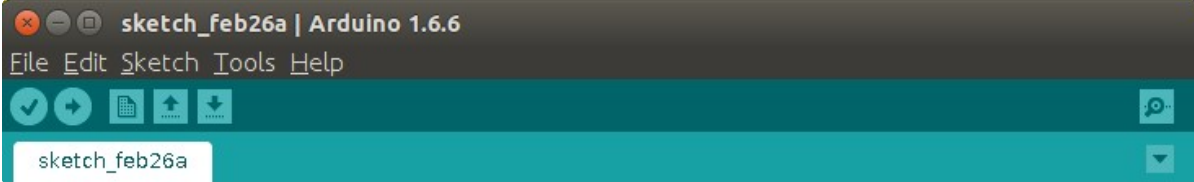
```

< >

111

Moving The Ball

Ping pong is no fun without the ball! We need to keep track of the position of the ball, it's speed, direction, whether it has collided with a wall, or a bat, and keep score. Phew, that's a lot! Remember though, we are *dividing and conquering* this problem, so we can add the code in small steps. In fact, we can even add it all in a single function! Have a look at lines 75 to 130 and try to work out which bits need adding.



```
1  #include <TFT.h>  // Arduino LCD library
2  #include <SPI.h>
3
4  // pins to control the screen
5  #define CS    7
6  #define DC    0
7  #define RST   1
8
9  // Pins for the player paddle positions
10 #define PIN_P1_BAT A2
11 #define PIN_P2_BAT A3
12
13 // create an instance of the library
14 TFT screen = TFT(CS, DC, RST);
15
16 const int bat_w = 5;
17 const int bat_h = 10;
18
19 const int ball_speed = 1;
20 const int ball_size  = 3;
21
22 const int screen_w = 320;
23 const int screen_h = 240;
24
25 // Player Scores
26 int score_p1;
27 int score_p2;
28 int position_p1;
29 int position_p2;
30
31 int ball_pos_x;
32 int ball_pos_y;
33 int ball_dir_x;
34 int ball_dir_y;
35
36 void updateBall();
37 {
```

```

38 // First, rub out the previous ball we drew.
39 screen.fill(255,255,255);
40 screen.stroke(255,255,255);
41 screen.circle(ball_pos_x, ball_pos_y, ball_size);
42
43 // Now let's update where the ball should be.
44
45 // Should it bounce of the walls?
46 if(ball_pos_y > screen_h)
47 {
48     ball_dir_y = -ball_speed;
49 }
50 else if (ball_pos_y < 0)
51 {
52     ball_dir_y = ball_speed;
53 }
54
55 // Should it bounce off player 1's bat?
56 if(ball_pos_x < bat_w &&
57     ball_pos_y > position_p1 &&
58     ball_pos_y < position_p1 + bat_h)
59 {
60     ball_dir_x = ball_speed;
61 }
62 // Should it bounce off player 2's bat?
63 else if(ball_pos_x > screen_w - bat_w &&
64     ball_pos_y > position_p2 &&
65     ball_pos_y < position_p2 + bat_h)
66 {
67     ball_dir_x = -ball_speed;
68 }
69 // Has player one scored?
70 else if (ball_pos_x > screen_w)
71 {
72     // increase player one's score
73     score_p1 += 1;
74     // reset the ball position to the middle of the screen.
75     ball_pos_x = screen_w/2;
76     ball_pos_y = screen_h/2;
77
78     // Where should we put the ball after a point is scored?
79 }
80 // Has player two scored?
81 else if(ball_pos_x < 0)
82 {
83     // .....
84 }
85
86 // Make the ball move in the right direction.
87 ball_pos_x = ball_pos_x + ball_dir_x;
88 ball_pos_y = ball_pos_y + ball_dir_y;

```

```

89
90     // Now draw the new ball
91     screen.fill ( // ..... What color shall the ball be?
92     screen.circle( // .....
93 }
94
95 void drawPlayerBats()
96 {
97     // Draw player 1's bat. First, clear the old area the bat took up.
98     screen.fill(255,255,255);
99     screen.stroke(255,255,255);
100    screen.rect(1,position_pl,bat_w,bat_h);
101
102    // Clear player 2's bat. How do we know the position of their bat.
103    // .....
104
105    // Read the player positions. This changes position_pl and position_pt
106    getPlayerPositions();
107
108    // Now we can draw the new bats
109    // Draw player 1's bat first.
110    screen.fill(0,255,0); // Make player 1's bat green.
111    screen.rect(1,position_pl,bat_w,bat_h);
112
113    // Now draw player 2's bat. Remember, it needs to be on the other
114    // of the screen.
115    // .....
116 }
117
118 void drawPlayerScores()
119 {
120     // This function will draw a very simple "X - Y" string where
121     // X and Y are the scores for each player.
122
123     // Set the text color to black.
124     screen.stroke(0,0,0);
125
126     // We will print five letters, so make our array five long.
127     char toPrint[5];
128     String S = "";
129     S += String(score_pl); // Add the player 1 score
130     S += " - ";           // Separate the two scores
131     // .....           // Add the player 2 score
132     S.toCharArray(toPrint,5); // Convert the string to something we can
133     // .....           // Display the text!
134 }
135
136 void getPlayerPositions(){
137     // Read player 1 position
138     // What do you think the map function does?
139     position_pl = map(analogRead(PIN_P1_BAT),0,1023-bat_h,0,screen_h);

```

```

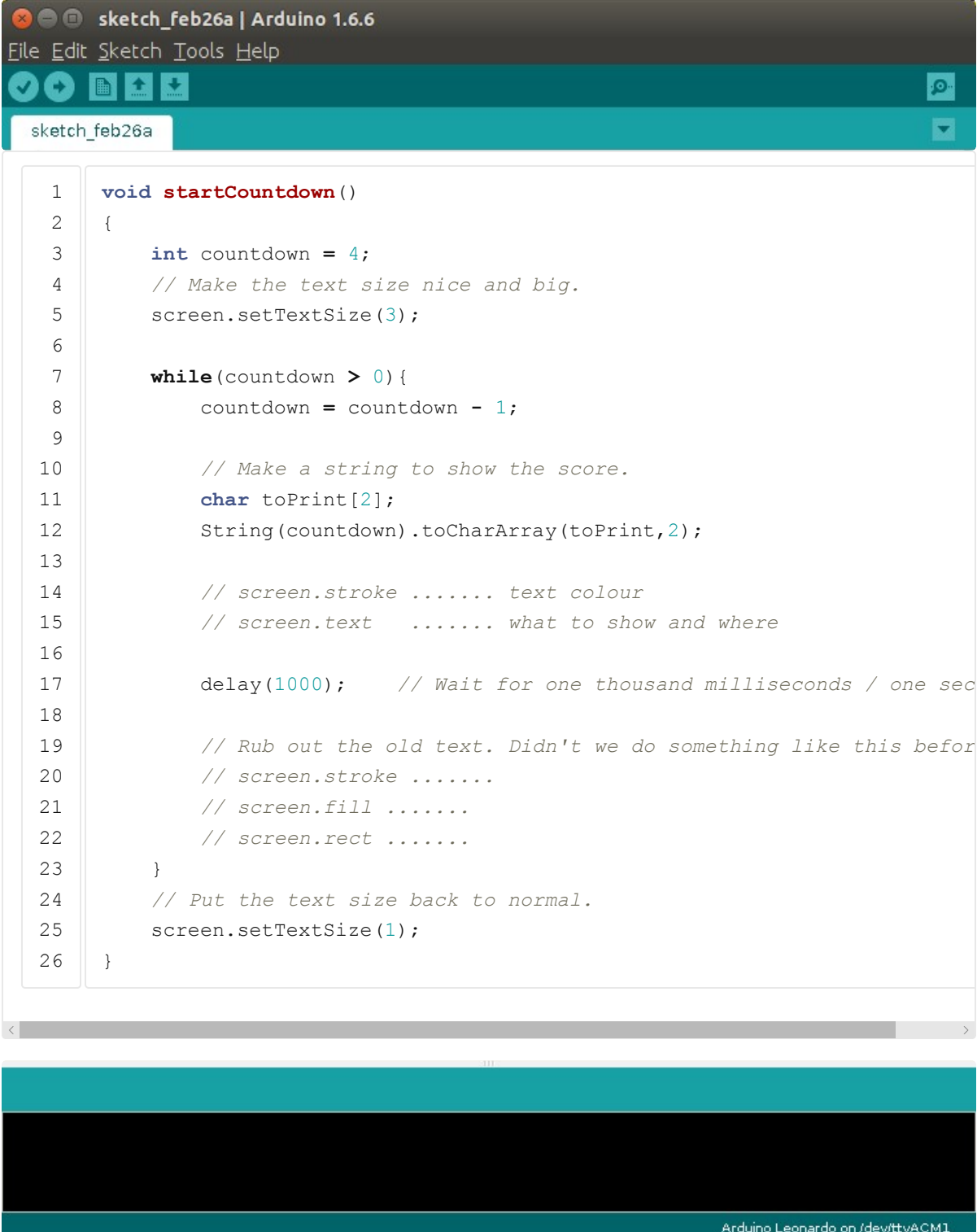
140
141     // Read player 2 position
142     // ....
143 }
144
145 void setup() {
146
147     // Set up the screen.
148     screen.begin();
149     screen.background(255,255,255);
150     screen.fill(255,255,255);
151     screen.stroke(0, 0, 0);
152     screen.setTextSize(1);
153
154     // Set up the player scores and bat positions.
155     score_p1    = 0;
156     score_p2    = 0;
157     position_p1 = 0;
158     position_p2 = 0;
159
160     // Set up the ball position and direction of travel.
161     ball_pos_x  = screen_w/2;
162     ball_pos_y  = screen_h/2;
163     ball_dir_x  = ball_speed;
164     ball_dir_y  = ball_speed;
165 }
166
167 void loop() {
168
169     // Move the "getPlayerPositions()" function call into the
170     // new drawPlayerBats() function. Can you figure out why?
171
172     // Draw the player scores.
173     drawPlayerScores();
174
175     // Draw the bats using the positions we read a moment ago.
176     drawPlayerBats();
177
178     // Draw and update the ball position.
179     updateBall();
180 }

```

Now, we are nearly there! We have the ball, the bats and a score counter. Now we just need to add a couple of functions to make starting and finishing the game a bit easier.

Winning!

So, at the moment, we are just thrust straight into the game and don't get any chance to get ready! Let's add a little count down to the start. There is some code below that displays a little count down from three. Can you work out where to put it in your code?



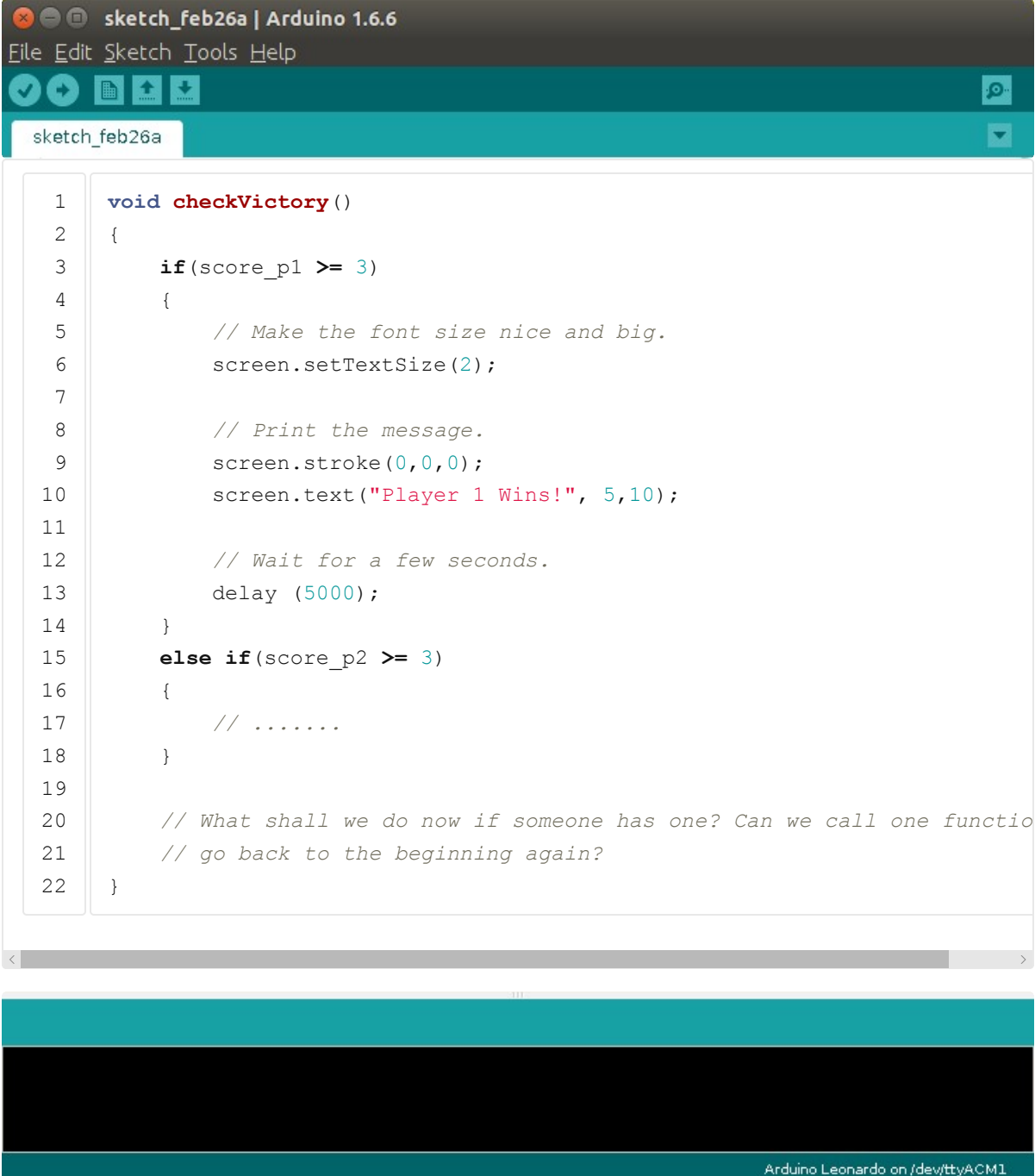
The screenshot shows the Arduino IDE interface. The title bar reads "sketch_feb26a | Arduino 1.6.6". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for opening, saving, and running a sketch. The sketch name "sketch_feb26a" is displayed in the top right. The main text area contains the following C++ code:

```
1 void startCountdown()
2 {
3     int countdown = 4;
4     // Make the text size nice and big.
5     screen.setTextSize(3);
6
7     while(countdown > 0){
8         countdown = countdown - 1;
9
10        // Make a string to show the score.
11        char toPrint[2];
12        String(countdown).toCharArray(toPrint,2);
13
14        // screen.stroke ..... text colour
15        // screen.text ..... what to show and where
16
17        delay(1000);    // Wait for one thousand milliseconds / one sec
18
19        // Rub out the old text. Didn't we do something like this before
20        // screen.stroke .....
21        // screen.fill .....
22        // screen.rect .....
23    }
24    // Put the text size back to normal.
25    screen.setTextSize(1);
26 }
```

At the bottom of the IDE, there is a status bar showing "111" in the center and "Arduino Leonardo on /dev/ttyACM1" on the right.

It might be nice to have this count down after each time someone scores. Can you make that happen by adding only two more lines of code?

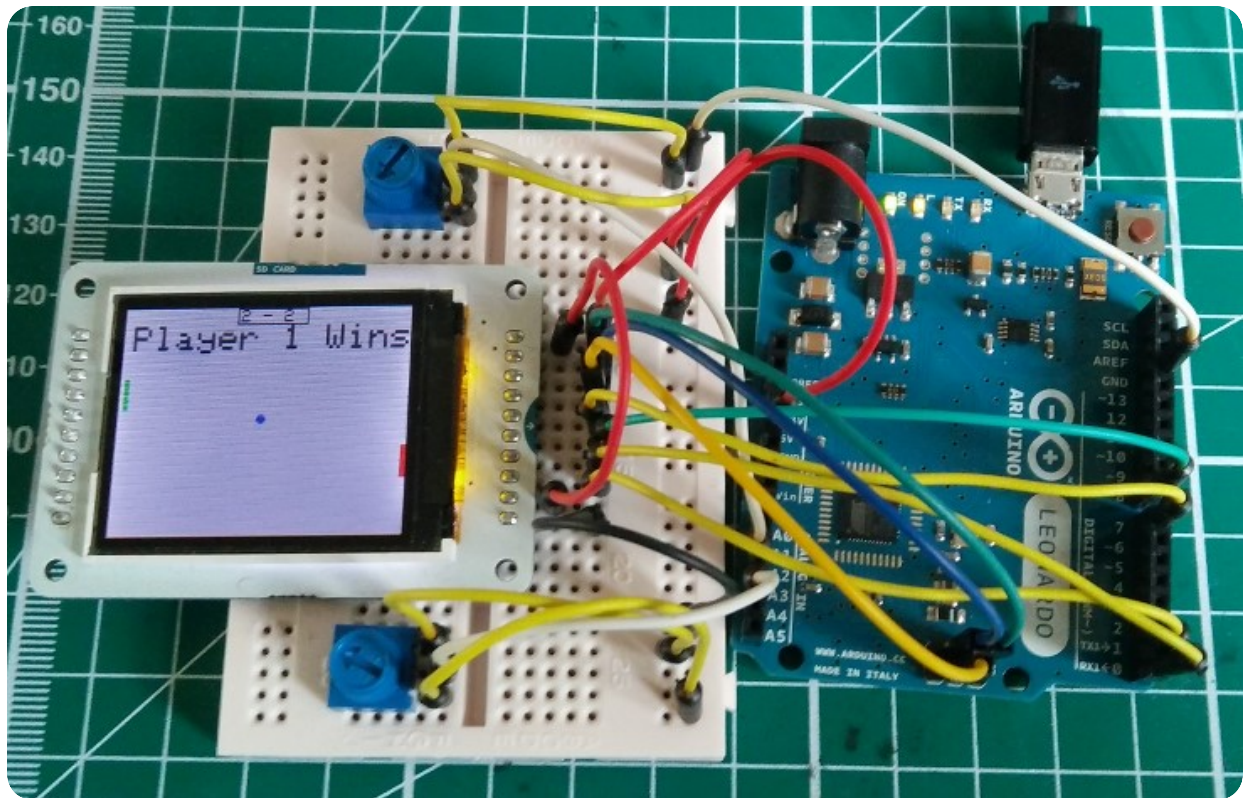
So, we can start the game nicely now, and play as well. We just need some sort of victory condition. Let's add one last function that checks if someone has won (first to score three maybe?) and show a message when they do. Again, can you work out where to put this function, and call it in your code?



```
1 void checkVictory()
2 {
3     if(score_p1 >= 3)
4     {
5         // Make the font size nice and big.
6         screen.setTextSize(2);
7
8         // Print the message.
9         screen.stroke(0,0,0);
10        screen.text("Player 1 Wins!", 5,10);
11
12        // Wait for a few seconds.
13        delay (5000);
14    }
15    else if(score_p2 >= 3)
16    {
17        // .....
18    }
19
20    // What shall we do now if someone has one? Can we call one function?
21    // go back to the beginning again?
22 }
```

The Finished Product

Congratulations! You've built your own Ping Pong game!



Since you found that so easy, can you make it better? How about using the buttons some how? Perhaps you think the colours are boring, or maybe you want to make another game entirely. This tutorial has told you everything you need to make awesome arcade games using the Arduino, so get hacking!

Be sure to post a picture of your creations to [our Facebook page](#) as well!

[< Buttons And Dials](#) | [Home](#) >

Arduino Games

Arduino Games

Written By [Ben Marshall](#)

bm12656@my.bristol.ac.uk

 [MVSE-Outreach](#)

 [digi_makers](#)

A tutorial written for the Digimakers project that teaches people how to build simple arcade games using the Arduino platform, an LCD screen module and other basic electronic components.