At The Conversation, we take table tennis *very seriously*. After setting up a new ping pong table, and wanting to learn how to use an Arduino, I had a bit of time to build a scoreboard.

An Arduino is a small, programmable device that can take inputs (in this case, two buttons) and control outputs (a LED screen to show the score).

The Arduino, and the components we used

The Arduino board is a relatively cheap one; at A$29, it’s a steal. It’s a great starting board, and easy to get your head around.

The LED matrix is a 32x16 display (512 LEDs in total). It provides a set of pins at the back to wire it up to the Arduino. The wiring documentation is a bit tricky to find, so I’ll save you the trouble
.

Nick Browne and I spent quite a while figuring out what buttons would be appropriate. Many were too small, or had a toggle state. Eventually, we settled on some that had the right *feel*
; we were after a tactile, satisfying click.

Additionally, we also bought a number of small components. These aren’t quite the ones we bought, but the links show equitable parts.

- a bunch of wires
- two five-metre 3.5mm headphone cables
- four 3.5mm headphone jacks
- two plastic enclosures to house the buttons

Prototyping the idea

When we started prototyping, we used some push buttons to act as inputs to the digital pins on

the Arduino.

During the ping pong game, each player pushes a button at their end of the table to increment their score after winning a point.

In the prototype, the buttons were a break in the circuit. Pushing the button connected the circuit, which we detected as the input.

The buttons are wired into a digital input pin and an analog ground pin. I've used pins 2 and 3, as you will be able to see in the code later on.

Wiring up the LED matrix to the Arduino was a challenge. The cable provided didn't work for us, and we think might have been faulty. We ended up using a bunch of individual wires.

Writing some C

Arduino is programmed in C, and runs a set of C/C++ functions. I've not written in a static-typed language before this project, nor a compiled one either, so this was new to me.

The Arduino looks for two specific functions, setup() and loop(). setup(), unsurprisingly, is run once and sets up the things you need for when the program is running.

Before you declare setup(), you can write a number of structs and constants.

All the data on a player, for example, can be written as a struct. Here's a cut down version of it:

typedefstructPlayer{intpin;intbuttonInputState;intbuttonState;intscore;}Player;

We also set the constants that measure what state the game is in.

The setup() is where we've set up all the default values: it sets player1pin and player2pin to inputs, and tells your Arduino to listen on certain pins. digitalWrite lets you listen for certain voltages, and so, listen for button presses.

Then, we're setting up the default values for the two player structs.

Here's a cut down example of the setup() method in the code.

```
voidsetup(){// initialize the pushbutton pin as an input: pinMode(player1pin,INPUT);pinMode(pl
ayer2pin
,
INPUT
);
digitalWrite
(
player1pin
,
HIGH
);
digitalWrite
(
player2pin
,
HIGH
);
for
(
int
i
=
0
;
i
<
NUMBER_OF_PLAYERS
;
i
++
)
{
players
[
```

```
i
].
buttonInputState
=
LOW
;
players
[
i
].
buttonState
=
UNPRESSED
;
players
[
i
].
score
=
0
;
// the last time the output pin was toggled
players
[
i
].
lastDebounceTime
=
0
;
}
}
```

We use constants to set the states, and the loop() progresses to the correct state once each small method is completed.

```
constintGAME_WAITING=0;constintGAME_START=1;constintGAME_ON=2;constintGAME_
OVER                                                                    =
3
;
```

The loop() is what runs continuously, and we’re using a variable named game_state to know which stage of the game we’re at. It then runs smaller methods that control each stage.

```
voidloop(){switch(game_state){caseGAME_WAITING:game_state=wait_for_players();break;c
```

```
ase
GAME_START
:
game_state
=
tell_players_its_on_like_donkey_kong
();
break
;
case
GAME_ON
:
game_state
=
game_on
();
break
;
case
GAME_OVER
:
game_state
=
game_over
();
break
;
}
}
```

The GAME_WAITING state scrolls text across the screen telling the players that pressing both buttons will start. It listens for the two simultaneous button presses, and if both pressed, sets the state to GAME_START.

The GAME_START state flashes START on the screen for five seconds, giving the players a chance to get ready! Once this timer is done, it sets the state to GAME_ON.

The GAME_ON state does the majority of the work in the program. It listens for buttons pushes and then increments the count. Then, it draws the scores on the LED matrix.

It keeps track of the scores, checking if either score is above 21, and if a player's score is more than two points greater than the other. If these conditions are met, it sets the state to GAME_OVER.

The GAME_OVER state figures out who won, and shows either P1 WINS or P2 WINS. Then, after waiting for a few seconds, it resets the state to GAME_WAITING.

I've made the source code for the Arduino available in a GitHub repository . Please take a look!

Wiring everything up

We simplified how buttons worked as inputs. We were able to remove the resistors and use the internal ones available on the Arduino's digital inputs.

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

The buttons keep the circuit connected, and pushing the button broke the circuit, which we used to detect the input.

We soldered everything together, adding a small prototype circuit board to organise the wiring from the Arduino to the two 3.5mm headphone jack pins.

From the headphone jacks, we ran two 3.5mm male to male cables. These connected to two other 3.5mm headphone jacks, fixed into plastic enclosures that also held the buttons.

I'm a bit of an amateur woodworker, so I made up a pine frame to fit the LED matrix panel. Four sides, each with 45 degree cuts, and some small brackets to fit them together.

A little sanding and varnishing, and it looks pretty good.

The finished product

With the LED matrix fitted into the timber enclosure, we're ready to turn it on and do some testing!

As you can see in the movie below, both players have to press their buttons simultaneously to start the game. Once it's started, each player pushes their buttons to increment their score after winning a point.

Once a player has more than 21 points, and has two more than their opponent, we have a winner!

Button pressing in action!

Authors: Mark Cipolla, Senior Developer & Content Platform Project Coordinator, The Conversation

Read more http://theconversation.com/how-we-built-an-arduino-powered-ping-pong-scoreboard-69401