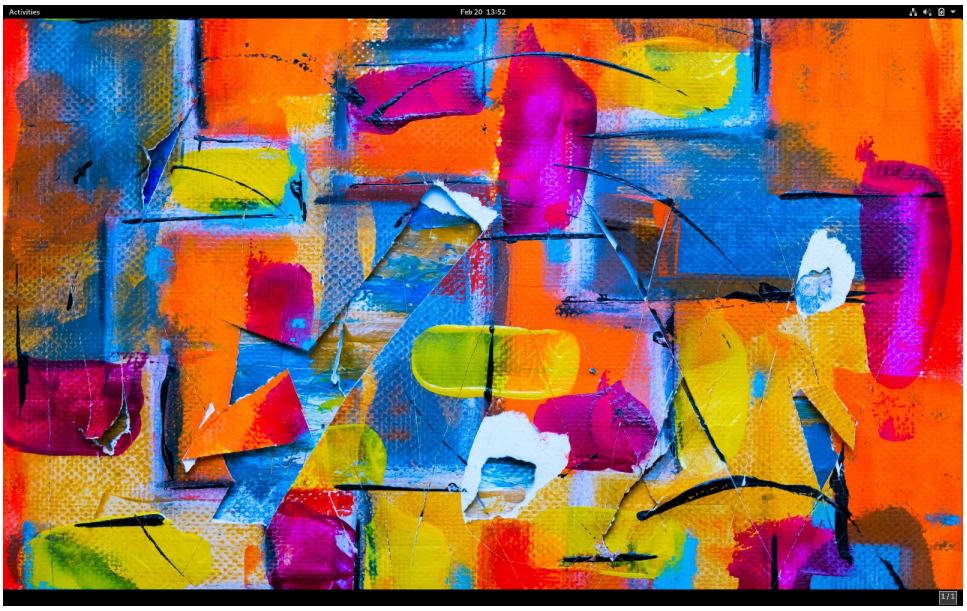
Embedded Software Workshop

Workshop Preparation

Steps:

1. Insert the provided USB stick into a USB port of your computer.
2. Turn off your computer, then turn it on and at the same time keep pressing on the F12 key 2-3 times a second to enter BIOS.
3. Select the USB option when asked what device to boot from.
4. Select the first option (Debian GNU/Linux Live) and wait for Linux to boot.

At this point you should see the following Debian desktop:



5. Click *Activities* (upper-left corner), search for *Terminal* and open it.
6. Before building a project, set the MAXIM_LIBRARIES, PLATFORM and TARGET environment variables with the following command:

```
$ export MAXIM_LIBRARIES=~/workshop/tools/MAX78000_SDK/Libraries; export PLATFORM=maxim; export TARGET=max78000
```

The *$* sign indicates that the terminal is ready to accept commands. Make sure NOT to include it in the command you are typing. Also keep in mind that the export needs to be run at every newly opened terminal.

Example 1:

This example prints a “Hello World” message over the serial UART.

Move to the no-OS workshop project location:

```
$ cd ~/workshop/no-OS/projects/workshop
```

Reset the workspace:

```
$ make reset
```

This command deletes the build directory (resulting in a fresh setup for starting the complete compilation process).

Build the first project example:

```
$ make EXAMPLE_NUMBER=1
```

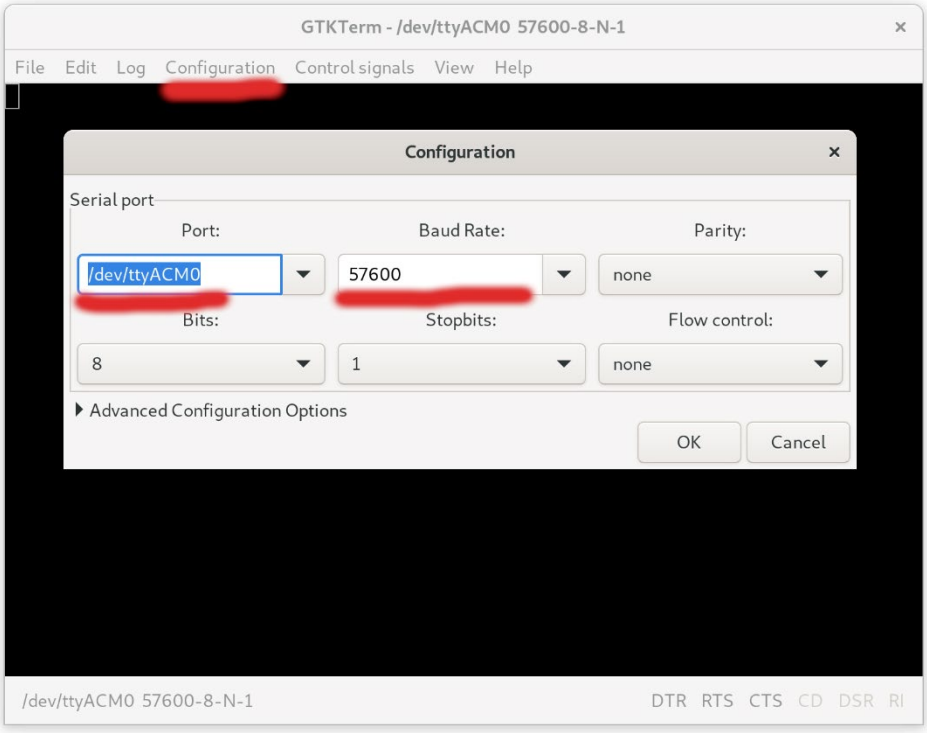
This command creates the build directory and the required directory structure, uses SDK (Software Development Kit) to create a project under the build directory and performs the build of files under the build directory using gcc, resulting in a binary file.

Connect the MAX78000FTHR to the PC using a USB cable.

To monitor the serial device of the computer we need a serial console application. Click *Activities*, search for *serial port* or *GTKTerm* and open the *Serial port terminal* application.

Configure GTKTerm application as shown below and click OK:

- Port: select the serial port on which the hardware device is connected through
- Baud rate: set the rate at which data is transmitted and received

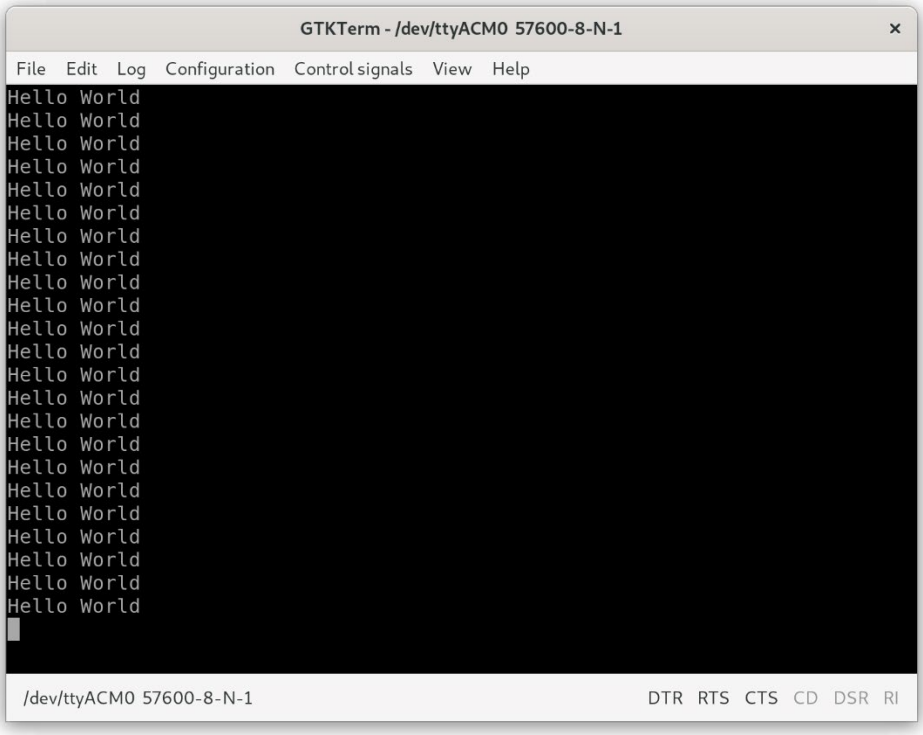


From the terminal, write the following command to program MAX78000:

```
$ make EXAMPLE_NUMBER=1 run
```

This command loads and runs the executable on the target board.

After the programming is completed, the GTKTerm application will print a “Hello World” message every second.



Example 2:

For this example, we connect an accelerometer to the microcontroller board using the provided wires. The scope of this example is to read the temperature from the accelerometer.

Disconnect the MAX78000FTHR USB cable from the PC.

Connect the EVAL-ADXL355-PMDZ to the MAX78000FTHR by using the information below:

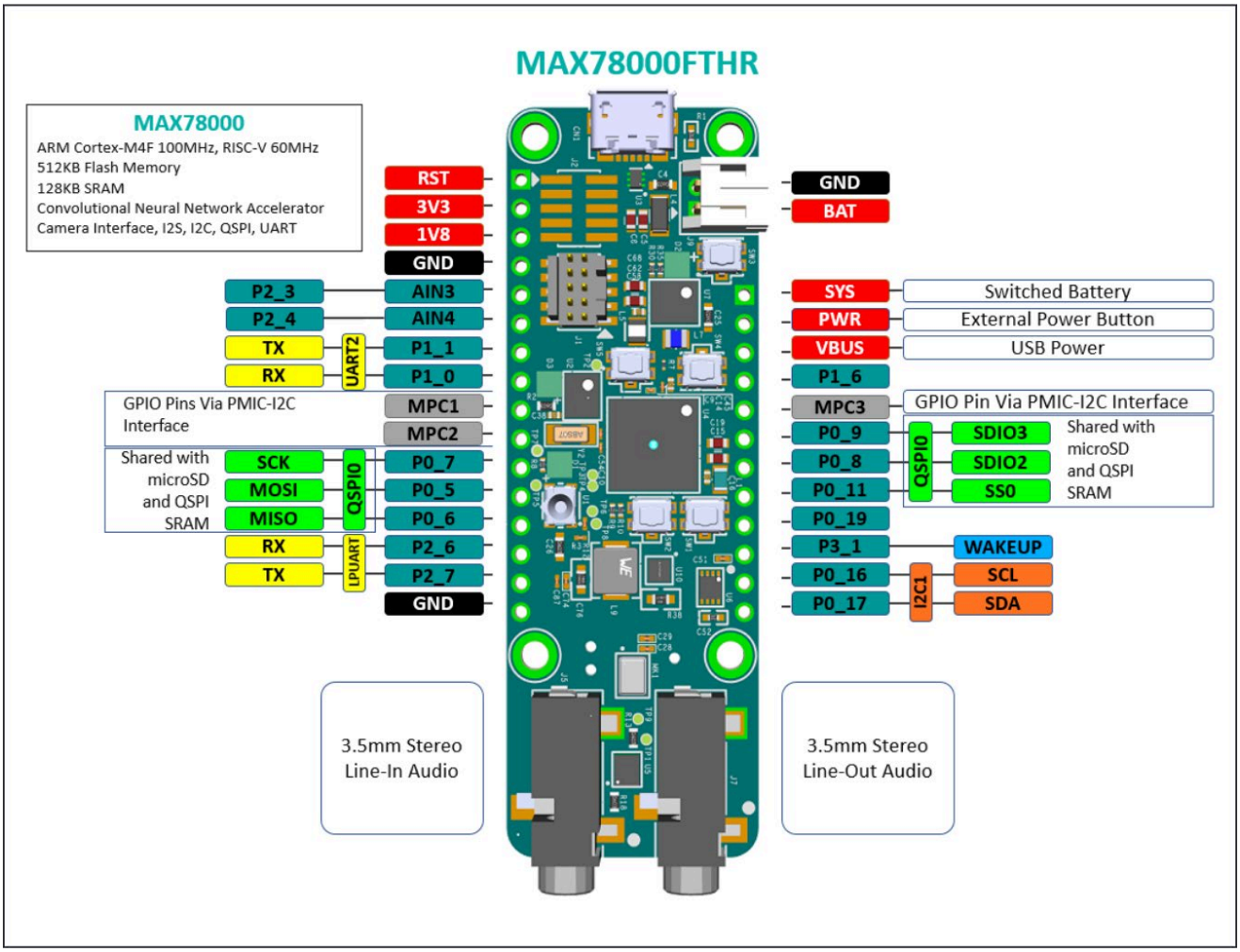
- EVAL-ADXL355-PMDZ Pinout
- MAX78000FTHR Pinout
- Pin correspondence table

EVAL-ADXL355-PMDZ Pinout:

Pin Number	**Pin Function**	**Mnemonic**
Pin 1	Chip Select	CS
Pin 2	Master Out Slave In	MOSI
Pin 3	Master In Slave Out	MISO
Pin 4	Serial Clock	SCLK
Pin 5	Digital Ground	DGND
Pin 6	Digital Power	VDD

Pin 7	Interrupt 1	INT1
Pin 8	Not Connected	NC
Pin 9	Interrupt 2	INT2
Pin 10	Data Ready	DRDY
Pin 11	Digital Ground	DGND
Pin 12	Digital Power	VDD

MAX78000FTHR Pinout:



Pin correspondence table:

MAX78000FTHR	Signal	EVAL-ADXL355-PMDZ
3V3	Digital power	6 or 12
GND	Digital ground	5 or 11
P0_11/SS0	SPI Chip Select	1
P0_5/MOSI	SPI Master Out Slave In	2
P0_6/MISO	SPI Master In Slave Out	3

P0_7/SCK	SPI Serial Clock	4

Make sure all 6 wires from *Pin correspondence table* are connected.

You may now plug in the MAX78000FTHR into the PC using the USB cable.

The accelerometer has an internal temperature sensor. This example makes use of this by reading it and displaying temperature values onto the serial terminal.

Make sure you are in the *~/workshop/no-OS/projects/workshop* directory and reset the workspace:

```
$ make reset
```

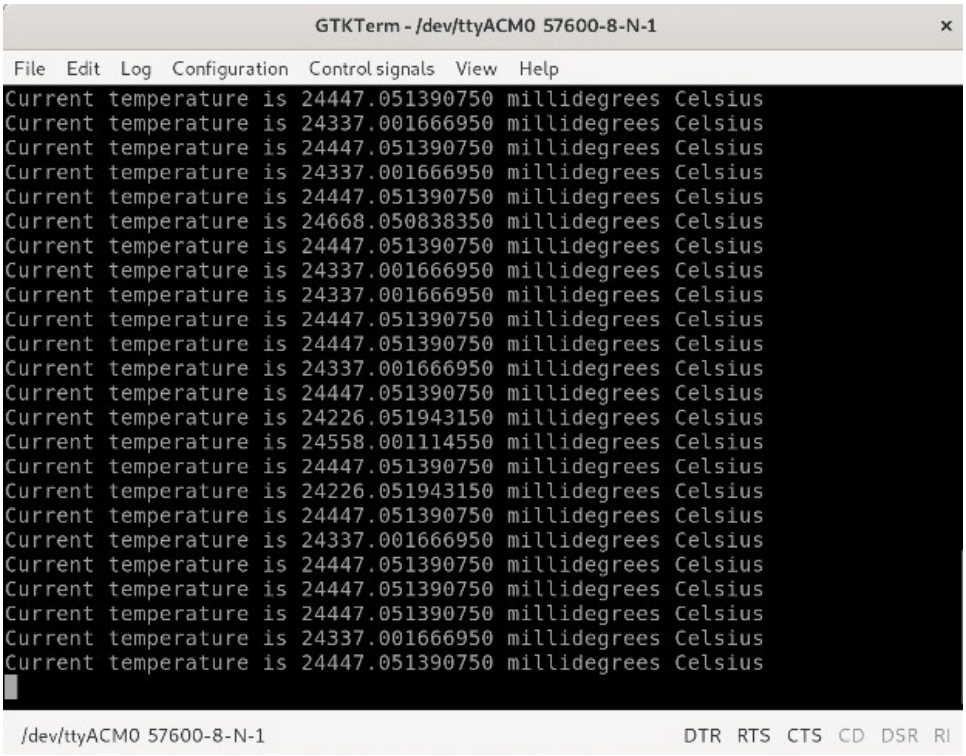
Build the example:

```
$ make EXAMPLE_NUMBER=2
```

Make sure GTKTerm is open and correctly configured and then load the example onto the board:

```
$ make EXAMPLE_NUMBER=2 run
```

Observe the output being printed every second.



CHALLENGE: change the current format of the printed temperature from millidegrees to degrees.

EXAMPLE: the current format: *27545.032056750 millidegrees*

the new format: *27.54 degrees*

HINT: the file you need to change can be found in *~/workshop/no-OS/projects/workshop/src* and is named *example_2.c .* After changing the file, you will need to reset, rebuild and reload the program onto the board.

Example 3:

Read temperature and the acceleration values from the ADXL355 and convert the data from raw values into user readable values.

Change the working directory and reset the workspace:

```
$ cd ~/workshop/no-OS/projects/eval-adxl355-pmdz
$ make reset
```

Build the WORKSHOP_EXAMPLE of this project:

```
$ make IIO_EXAMPLE=n WORKSHOP_EXAMPLE=y
```

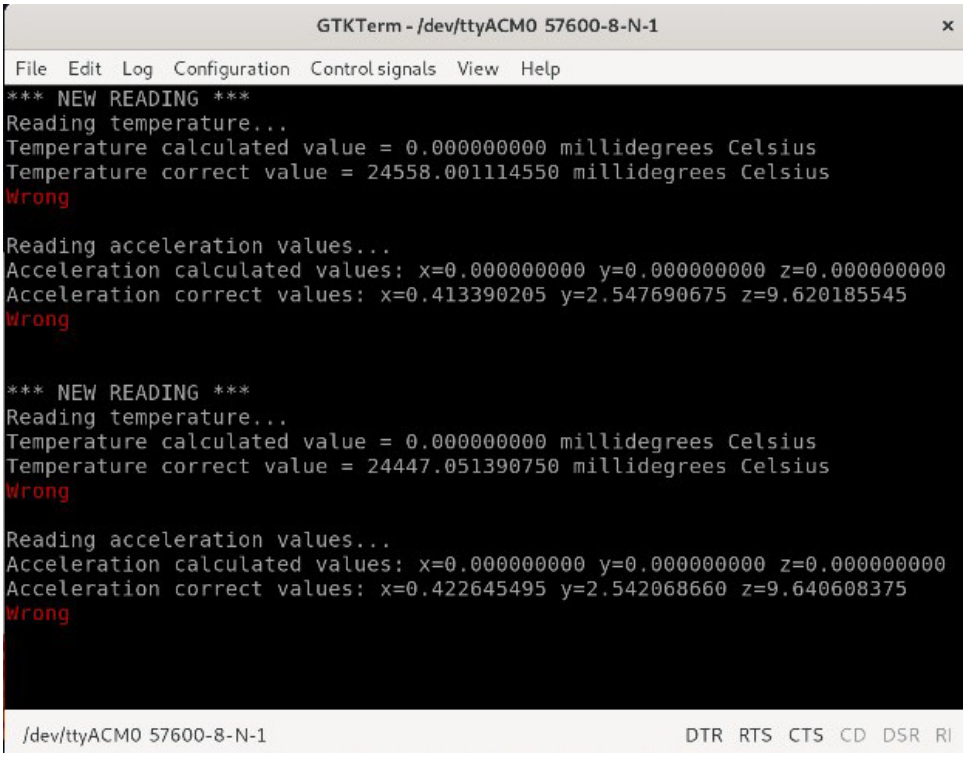
Make sure GTKTerm is open and correctly configured.

Load the example onto the board:

```
$ make IIO_EXAMPLE=n WORKSHOP_EXAMPLE=y run
```

Notice the output format:

- The calculated value of the temperature
- The correct value of the temperature
- *Wrong* message that indicates that the values do not correspond
- The calculated values of the accelerations
- The correct values of the accelerations
- *Wrong* message that indicates that the values do not correspond



CHALLENGE: compute the temperature and the accelerations from the raw values.

HINT: For temperature you need to compute the *temp_dividend* and *temp_divisor.*

For accelerometer values you need to compute the *x_dividend, y_dividend, z_dividend* and *accel_divisor.*

The file you need to change can be found in *~/workshop/no-OS/projects/eval-adxl355-pmdz/src/examples/workshop* and is named *workshop_example.c .* After changing the file, you will need to reset, rebuild and reload the program onto the board.

The formula for the temperature:

$$TEMPERATURE = (RAW + OFFSET) \cdot SCALE$$

$$TEMPERATURE = \left(RAW + \frac{OFFSET}{OFFSET\_DIV}\right) \cdot \frac{SCALE\_FACTOR}{SCALE\_FACTOR\_DIV}$$

$$TEMPERATURE = \frac{(RAW \cdot OFFSET\_DIV + OFFSET) \cdot SCALE\_FACTOR}{OFFSET\_DIV \cdot SCALE\_FACTOR\_DIV}$$

The formula for the acceleration:

$$ACCELERATION = RAW \cdot SCALE$$

$$ACCELERATION = \frac{RAW \cdot SCALE\_FACTOR\_MUL}{SCALE\_FACTOR\_DIV}$$

Parameter correspondence table:

PARAMETER	VALUE
TEMPERATURE OFFSET	- 2111.25
TEMPERATURE SCALE	- 110.497238
ACCELERATION SCALE	0.00003824593

Macro correspondence table:

MACRO	VALUE
ADXL355_TEMP_OFFSET	- 211125
ADXL355_TEMP_OFFSET_DIV	100
ADXL355_TEMP_SCALE_FACTOR	-110497238
ADXL355_TEMP_SCALE_FACTOR_DIV	1000000
ADXL355_ACC_SCALE_FACTOR_MUL	38245
ADXL355_ACC_SCALE_FACTOR_DIV	1000000000

To compute the temperature and the accelerations you can use either the macros or their values from the above table. The *ADXL355_TEMP* macros correspond to the temperature values, and the *ADXL355_ACC* macros correspond to the acceleration values.

The raw values from the formulas can be found in the source file under the name raw_<temp, x, y, z>.

OBSERVATION: for the division to be correct, the dividend must be represented on twice the number of bits the divisor is represented on, so you need to cast explicitly one parameter of the dividend to *int64_t*. Example: *(int64_t)ADXL355_TEMP_SCALE_FACTOR.*

Example 4:

This example consists of an accelerometer-enabled game that lets you place components on a circuit by physically tilting the accelerometer.

Close the GTKTerm.

Make sure you are in the *~/workshop/no-OS/projects/eval-adxl355-pmdz* directory and reset the workspace:

```
$ make reset
```

Build the IIO_EXAMPLE of this project:

```
$ make IIO_EXAMPLE=y WORKSHOP_EXAMPLE=n
```

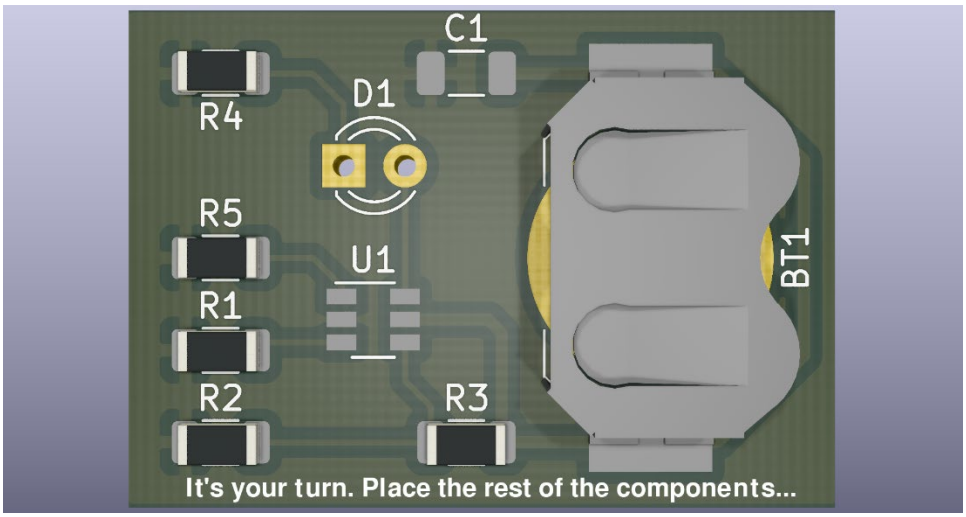
Program the board:

```
$ make IIO_EXAMPLE=y WORKSHOP_EXAMPLE=n run
```

Change the directory and run the game:

```
$ cd ~/workshop/play
$ python3 play.py
```

Notice the graphical interface:



Move the accelerometer board around and observe the output. Be careful not to disconnect the wires.

Let’s see if you can beat the game!