TTISD

Bjorn Jorissen, William Thenaers

Assignment 1: Android geoTODO

Some screenshots from an emulator can be found in `TTISDassignment1\doc\`.

A video showing the execution on a real Android smart phone and a Samsung Gear Live is also inlcuded.

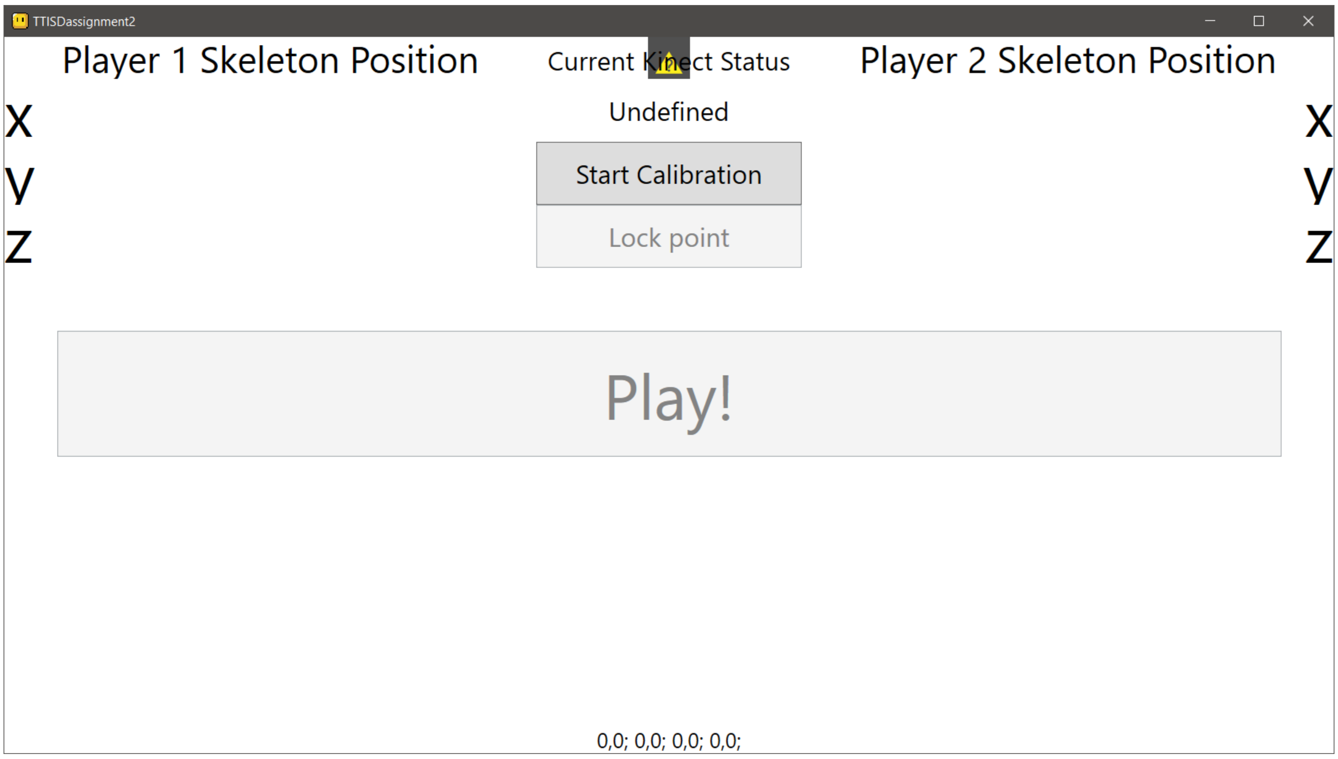
Assignment 2: Interactive Room

Some screenshots and a video can be found in `TTISDassignment2\doc\`.

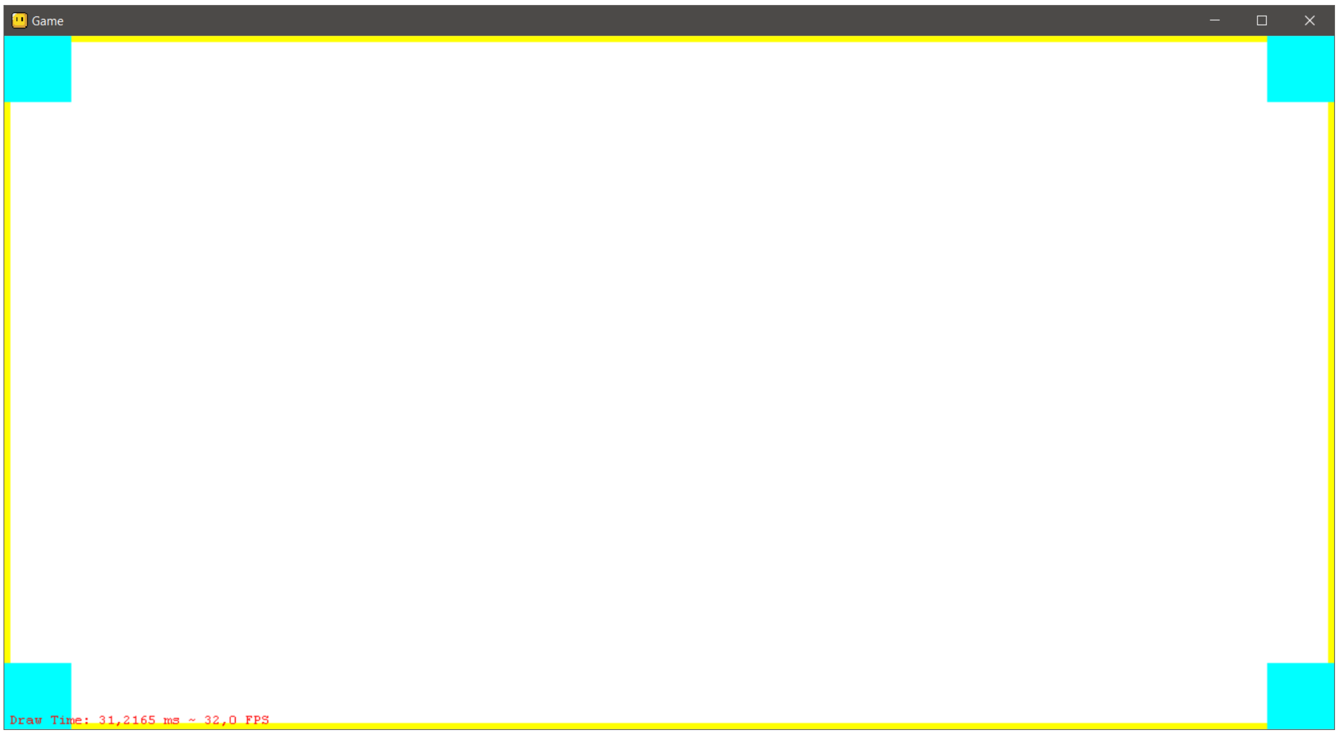
A release build is provided in `TTISDassignment2\bin\Release`.
Running it might require the Kinect SDK (v1.8).

Gameplay

First, someone presses the `Start Calibration` button.

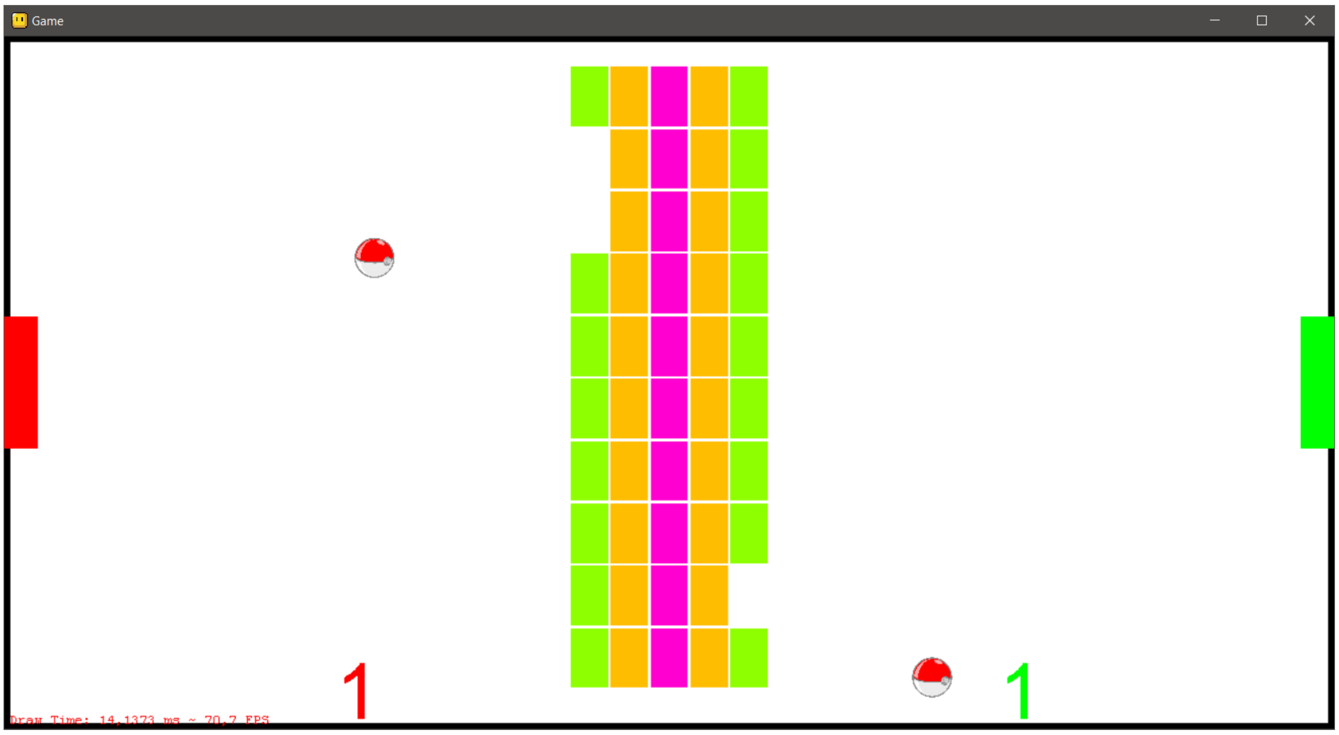


Four cyan square will appear in each corner of the game window. When starting calibration, only one will be shown at a time. The tracked skeleton position of one person will be shown on the left side of the above status window. The person should stand on a projected square, and pressing `Lock Point` will calibrate the game corner the person is standing on with their current Kinect skeleton position. After locking, the next square will be shown, until all points are locked.



After all four corners are calibrated, a second person can join the Kinect viewing range and when the two people are being tracked (both XYZ positional values are being updated in the status screen), someone can press play and the game will start.

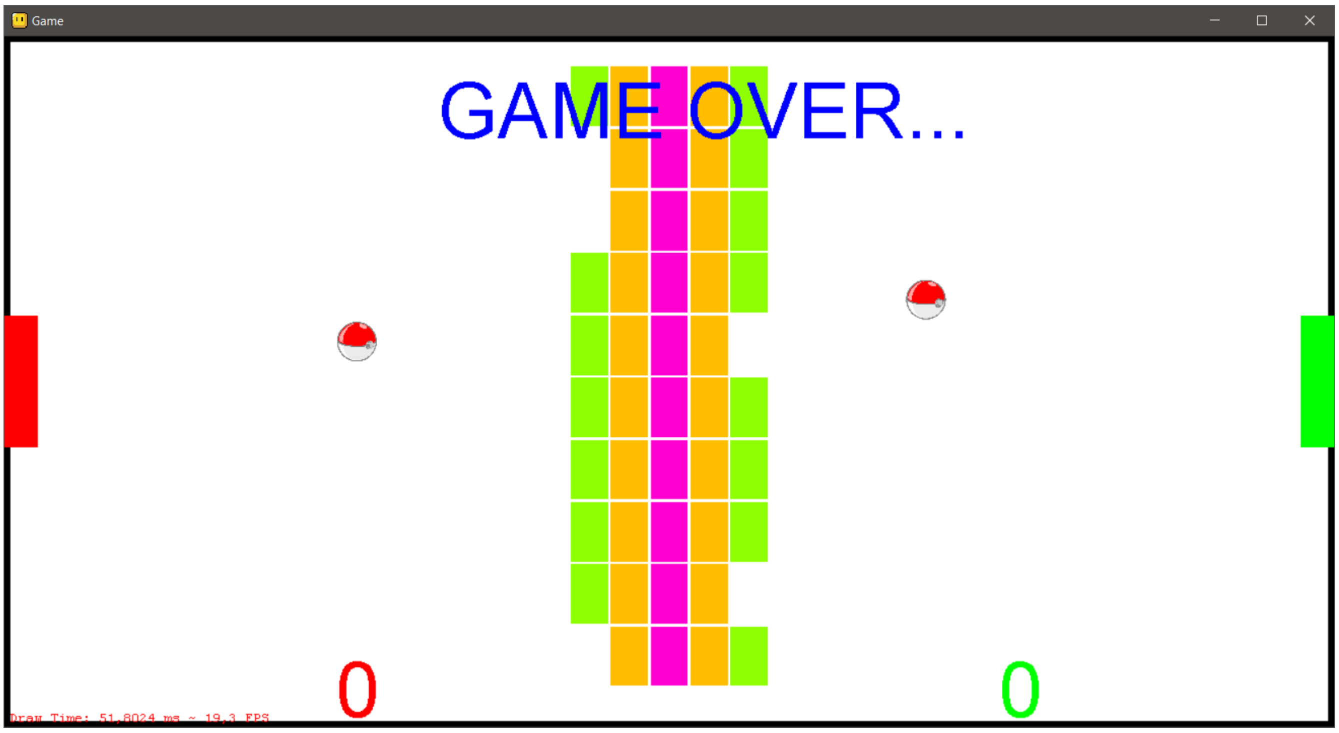
The purpose of the game is gathering as many points as possible by destroying the blocks in the middle, and when they are destroyed, players must compete against each other to bring the other's score to 0. The last player with a score greater than 0 wins.



The colour of a block determines the score a player gets by destroying it (green gives 1 point, yellow 2, orange 3, up to magenta with 5 points). Hitting a block will transition it to a lower state, e.g. hitting a magenta block will give the player 5 points, then the block will turn red and will give 4 points on the next hit. A green block worth 1 point will disappear when hit.

The last player that touched the ball will get the points. If a ball reaches the left or right boundary on a player's side, 5 points will be deducted from that player and the ball's position and speed will reset to the side the ball was lost in. The ball will start moving towards the other side. When a player hits the ball, its speed will slightly increase, while hitting a block will reset the speed modifier.

If all blocks are destroyed and one player loses all his points, then the other player wins. If both players lose all their points (by not preventing a ball from reaching their border), the game ends in a draw as seen below.

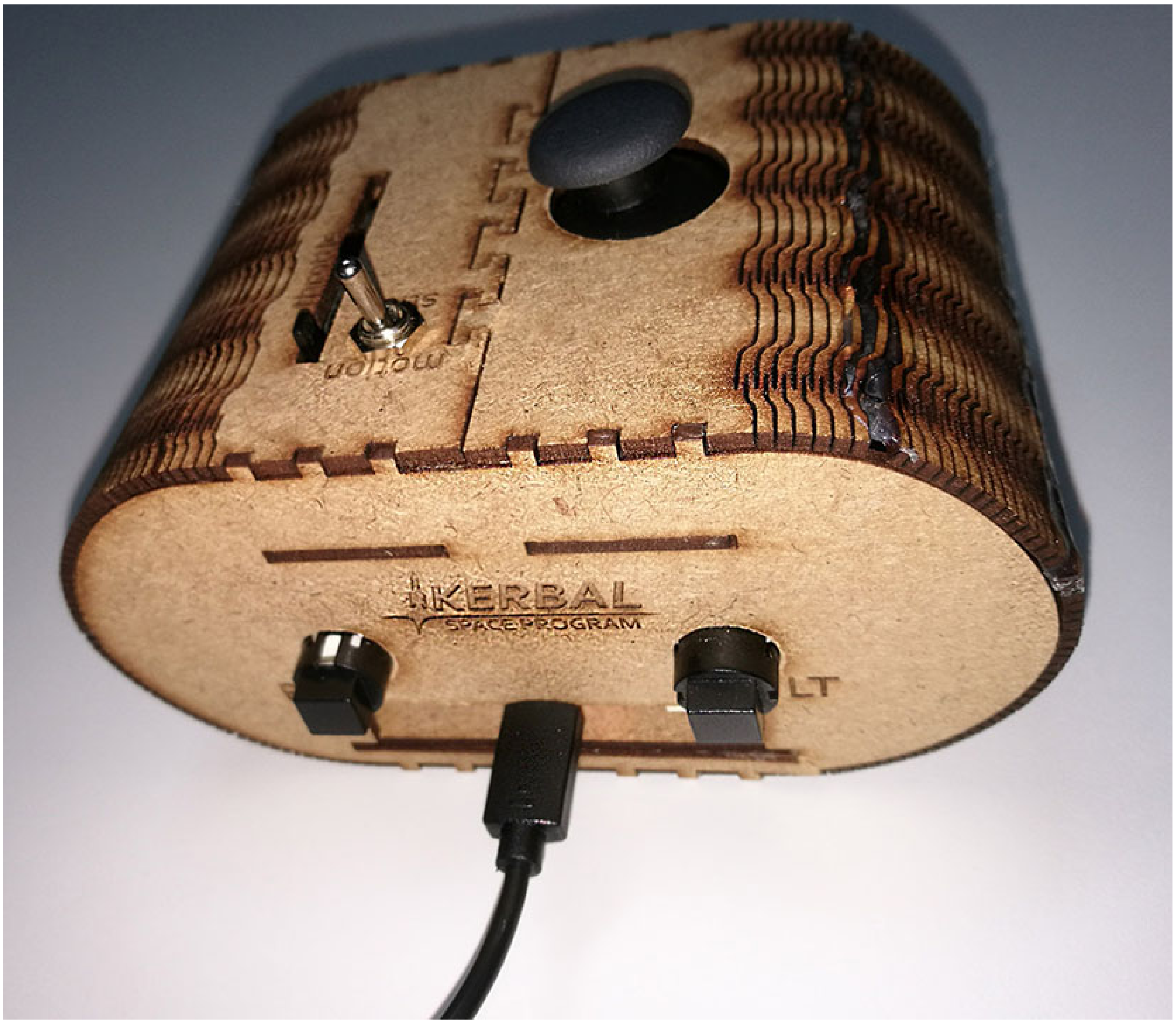


```
Draw Time: 51,8024 ms ~ 19,3 FPS
```

Assignment 3: Gamepad controller

We made a game controller for [Kerbal Space Program](#) (KSP), with the following features:

- Control your craft (pitch, yaw, roll) by moving the controller or the joystick around;
- Adjust the thrust output (throttle);
- Switch between motion or joystick control mode.



Code

The Microbit runs a simple main loop that collects data from each of the following inputs:

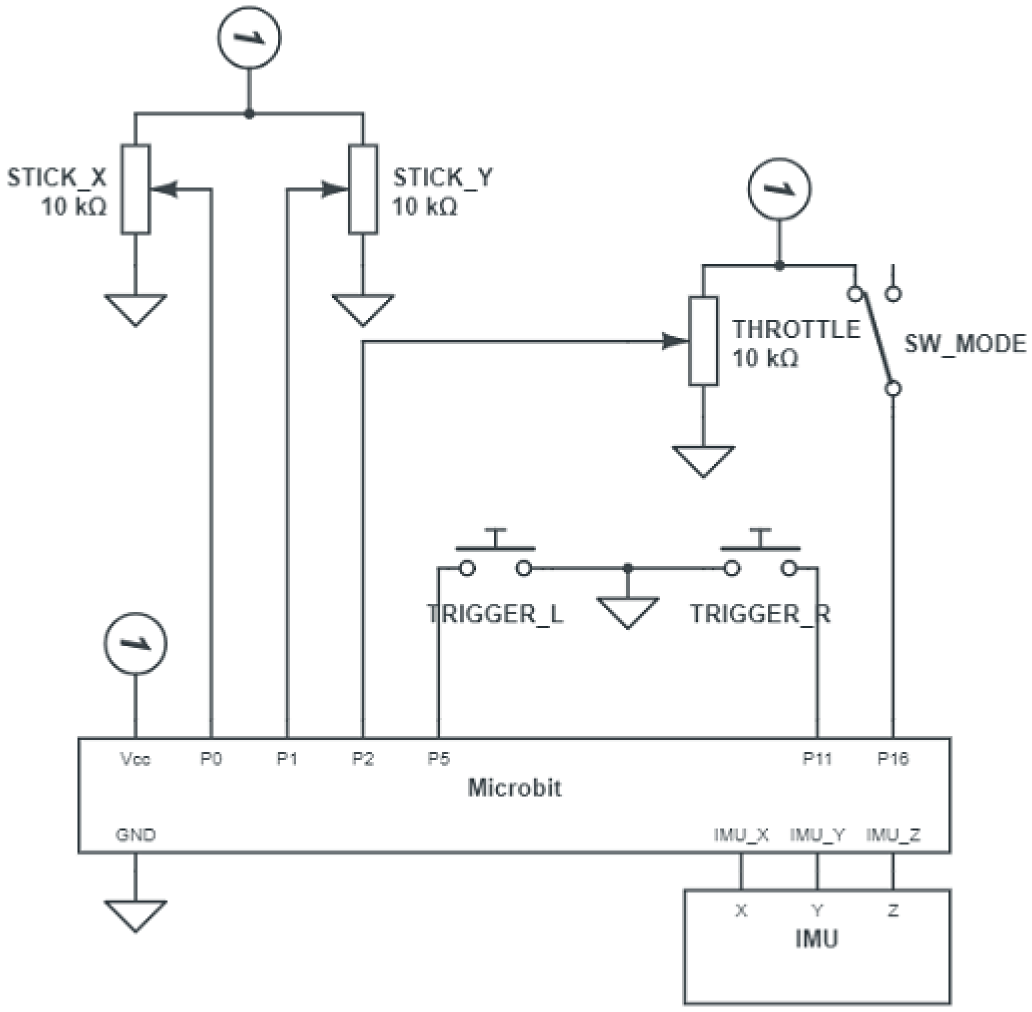
- The joystick (analogue X/Y-axes) for pitch and roll control;
- The Inertia Measurement Unit (analogue X/Y/Z-axes) for pitch and roll control;
- Two trigger buttons for yaw controll;
- A linear potentiometer for amount of thrust;
- A switch to toggle joystick or IMU control mode.

Since the assignment required Python, both the Microbit and the host computer run a script, `Code/KSPCommander_device.py` and `Code/KSPCommander_host.py` respectively. Installing the `PySerial` package on the host is required.

By using MicroPython on the Microbit, there is not enough memory left to use the full `radio` module for Bluetooth support, and hence we had to give up on using it. Since the working memory is also limited when using Python, the script had to be split between the Microbit and the host, else the entire interaction with the Microbit and KSP could have been contained on the Microbit only, We used the KerbalSimpit mod (open source) and adapted its companion Arduino library for the Microbit. After handshaking, the mod handles packets sent from a serial port to adjust data in the game. Since the code could not be contained on just the Microbit, the additional host script requires an emulated COM port to communicate with KerbalSimpit. This is achieved with the external program com0com (install can be found in `Code/deps`).

Enclosure

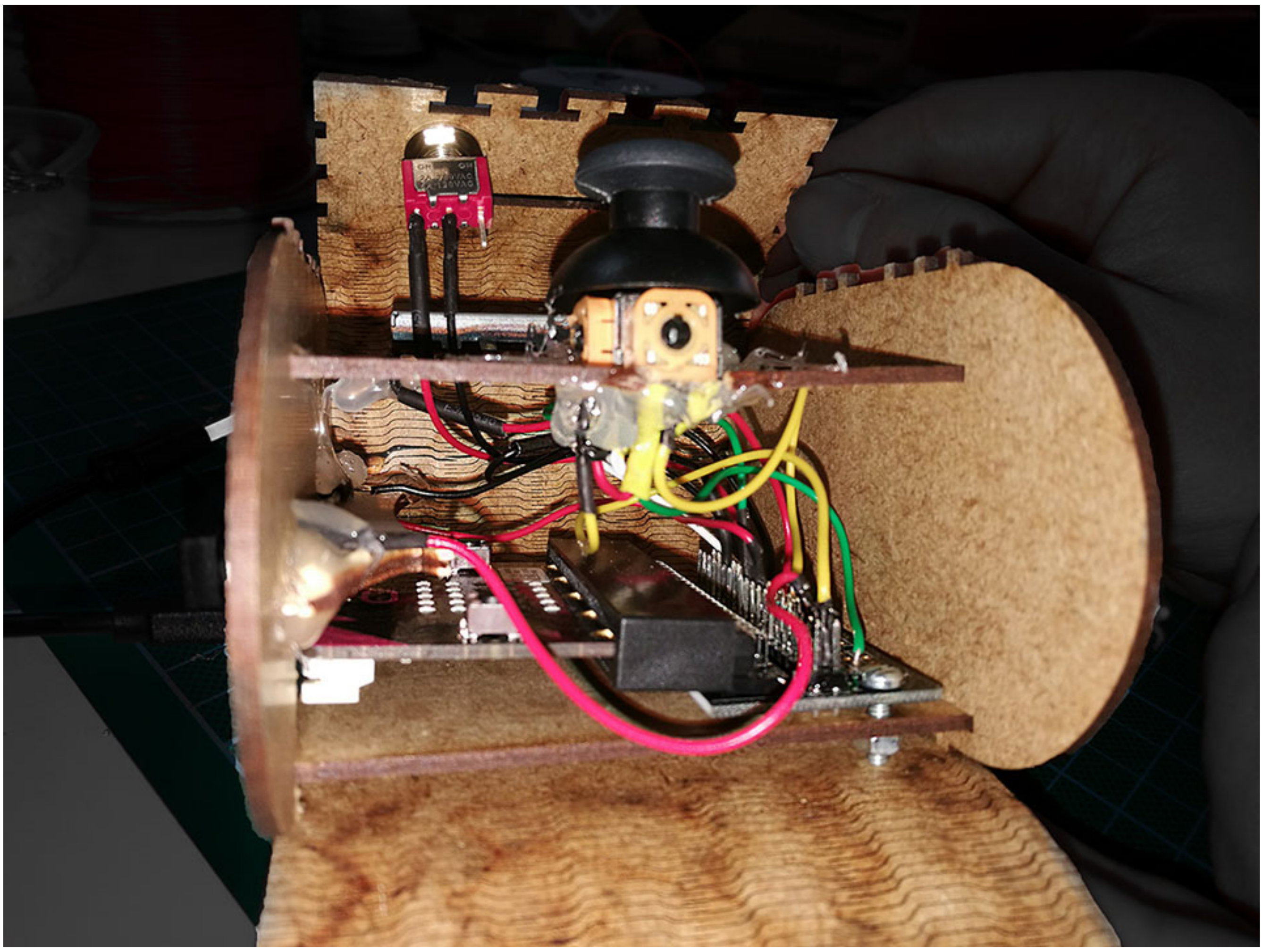
Refer to the circuit diagram and the SVG drawing in the `Enclosure` folder.



The triggers are connected to the same pins as the on-circuit buttons A and B, and thus don't require an extra resistor. The mode switch does not require a resistor as well, because digitally read pins are automatically pulled down internally. So connecting to Vcc provides a clean reading. The other inputs are resistors themselves (stick and throttle slider) and are not pulled as they are analogue.

The enclosure was cut from a 2 mm MDF sheet. Both sides are living joints to provide a better grip when holding the controller.

The wiring is directly soldered to the extension board, covered with heat shrinks where applicable.



Gameplay

Before starting, check on which port the Microbit is connected and make sure to install com0com (the default COM to COM link is fine).

1. Open `KSPCommander_host.py` and adjust the `MicroPlayer` arguments under `__main__` accordingly:
    - `device_serial_port` should be the COM port where the Microbit is connected on;
    - `ksp_serial_port` should be one of the 2 virtual COM ports created with com0com.
2. Install KSP and use CKAN (KSP mod manager) to install KerbalSimpit;
3. Navigate to the KSP install folder and go to `Kerbal Space Program\GameData\KerbalSimpit\PluginData`. Rename `Settings.cfg.sample` to `Settings.cfg` and change the `PortName` variable to the other COM port from the virtual pair (as in the script uses one, and KerbalSimpit will use the other);
4. Start KSP, plug the Microbit in the computer and start the host script;
5. When ready, press both triggers on the controller at the same time, the host script will try to handshake with the mod. When this succeeds a message will be displayed in the console;
6. Choose or build a vessel to fly and mod will use the data from the controller to override the keyboard controls. You can now control your flight with the controller!

A video showing the handshake setup and some gameplay can be found in `Doc`.