LiDAR_PONG

Age Group
All ages (5+)

Background
Pong is a classic 70s Atari/Arcade game - one of the first video games ever made. This project is intended to feel similar to an arcade cabinet, however operates using lidar sensors instead of a joystick. The position of the player's hand from the sensor corresponds relatively to the position of the paddle on-screen. It is designed to be a 2 player interactive game that demonstrates the basic concepts of LiDAR / Time of Flight.

What to do – Uploading/Updating Code
The board in use is the Arduino DUE. The easiest method to get the unit operational is by using the Arduino IDE. There are two versions, IDE 1.X.X and IDE 2.X.X. I would recommend using IDE 2 and so the steps detailed here are specific to IDE 2.

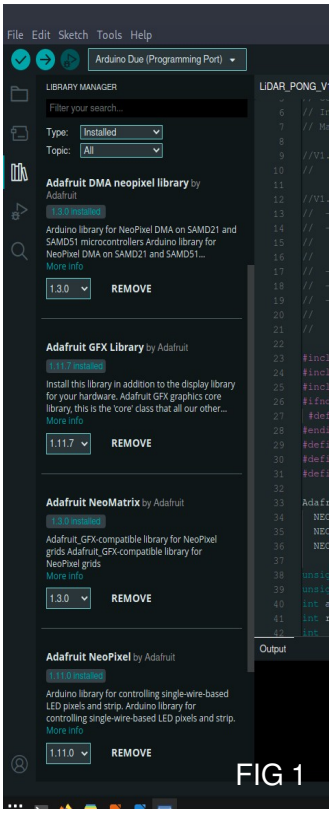


FIG 1

Once installled and operational, install the following libraries from the library manager (as shown in Figure 1). [Adafruit GFX, Neopixel, Neomatrix + allow auto install of all dependencies]

You will also need to add the Arduino DUE to the boards manager – See Figure 2.

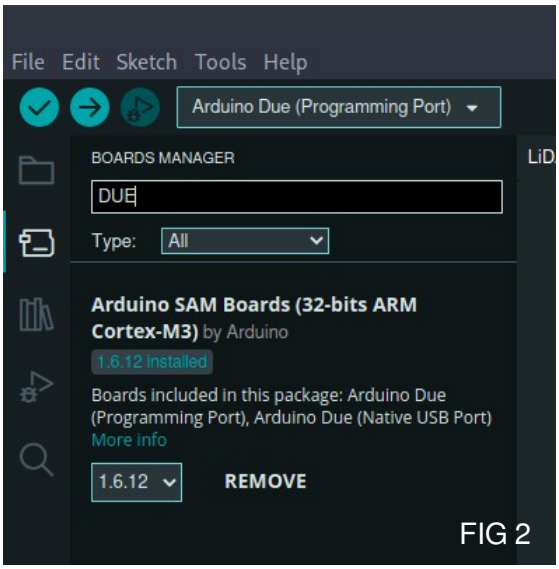


FIG 2

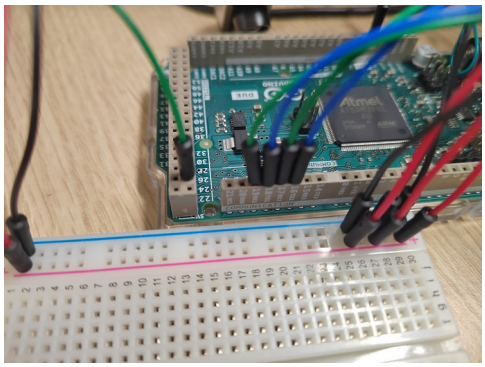
HOST INSTITUTION
Tyndall National Institute
PARTNER INSTITUTIONS
UCC
DCU
CORK INSTITUTE OF TECHNOLOGY
FUNDED BY:
European Union
Science Foundation Ireland

Open the LiDAR_PONG_V1.X.ino code file in the Arduino IDE. If the code file is lost or unavailable, the file can be re-obtained/downloaded from the following repository: https://github.com/saltycmd/LiDAR_PONG – NOTE: The version showcased at the presentation in August 2023 was V1.0. The version I recommend using is V1.1. I plan to update the software over time with new features if I ever build my own unit! You are welcome to use newer versions.

Connect the Arduino DUE to your computer using the 'programming' usb port (This is the one closest to the black DC in connector). Upload the code file to the arduino by selecting the board in board selector, located just under the 'Tools' tab on the top function bar – then click the upload button located to the left of the selector (Arrow pointing to the right).

Electronics

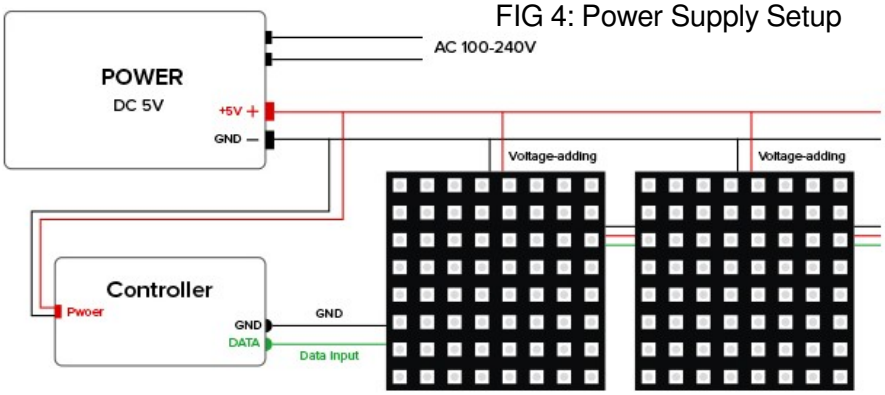
The unit can be powered by one of two methods – via USB ports located on the board (It should not matter which USB port on the board you use for code V1.1) or by a DC adapter to power the Arduino. Due to a limitation of wire colours, the connections in their current state are not fully colour co-ordinated. The lidar sensors communicate using RS-232 Serial. A general rule of thumb to setup the lidar sensors is: RED → +5V, BLACK/WHITE → GND, GREEN → RX, BLUE → TX. For the given code files these should be connected to the Arduino as shown in Figure 3.



The matrices communicate via digital (DIN). This is just one connection: GREEN → Pin 22. The matrices are powered by RED → +5V, BLACK/WHITE → GND. Figure 4 shows how the matrices should be setup and powered if using the external power supply.

FIG 3: Arduino Connections

FIG 4: Power Supply Setup



Gameplay

Once the Arduino is powered and electrical connections have been correctly configured, the game will boot into a menu screen. Players can adjust to the mechanics/controls at this point. **The game is started by having both players bring their hand close to the sensor and allowing the game to draw in the boundaries.** NOTE: At any point if a player has their hand too close to the sensor the game will freeze until sufficient distance between hand and sensor is detected. The flashing red arrows in the games menu is a rough indication of where the players are expected to hold their hands until the boundary is drawn in. Once the boundary is drawn in, the game starts immediately.

Additional Information

- At any point if a player has their hand too close to the sensor the game will freeze until sufficient distance between hand and sensor is detected.
- The unit, by rite, should be powered by the power supply purchased. This is something we did not have time to setup, however the game is in a very functional state. However, there is what I believe to be a direct impact on gameplay because of this. Since the current setup is powered soley via USB, and matrices are powered through the Arduino, the game freezes. I believe this is an over-draw of current from the Arduino to the matrices, causing it to freeze but not turn off. My claim us supported by the fact that the freezing happens more regularly at higher LED brightness settings. I am 99% sure that the code is not at fault for this.
- There is two modes that have been made, an ‘easy’ mode for younger children and a ‘competitve mode’ in which the speed of the ball gets more rapid with every 2 collisions. Currently this need to be toggled manually in the code – the 3d model provided has left cut outs for switches to switch between modes. This will require mode code which I would be happy to help with.
- The winner is the player that gets a score of 7 first, there is no ties.

Contact

jmcc097@gmail.com

120387836@umail.ucc.ie

Code Files

https://github.com/saltycmd/LiDAR_PONG