CS6650 Smart Sensing for IoT Diwali '21

Final Report

Group Information:

No.	Team Member Name	Roll. No.	Contribution
1	Akshat Meena	CS19B052	41%
2	Vedaant Arya	CS19B046	36%
3	Mohammed Shamjid AT	EE21Z042	19%
4	Yash Bhagwat	EP19B021	4%

I. USAGE

To begin, simply plug the board into a power source. Unpowered USB hubs are not recommended since current draw may be higher than supplied.[1]

The board is now active and will look for a host. It is useless on its own.
Connect to the board via your computer's Bluetooth Settings.[2]

In the provided script, change the constant SER_PORT to the serial port. This may be "COM0" or similar for Windows, or "/dev/rfcomm1" or "/dev/ACM0" or similar for Linux.

Run the script **`main.py`**, using python3. You might require **`matplotlib`** and **`numpy`**.

It will enter the *calibration* mode. The script will guide you as you go along. Kindly follow the instructions. It was intended that parameters are not stored.
Once it is done, the script will begin sensing and show the sensed and digitized touch on a plot, it shows the previous 5 instances.[3]

1. main variant uses 5 LEDs which each draw 20mA expected, however with the HC-05 having a max draw of 40mA may exceed the 100mA supply.
2. you might need additional setup to ensure that a Serial Port is created and available for the corresponding Bluetooth device (HC-05).
3. tweak the parameter PLT_TRAIL to change the number of instances to show on the graph.

II. PROPOSAL

We planned to use IR sensors to track movement of a single pointing object on a surface. Basic idea is to make a system where we can mimic writing some patterns (e.g., geometrical shapes, alphabets) that the IR sensing system can identify and decode. We made the following two variants:

1. **Single Bar** - IR sensor and sender arrays will be placed at one side of the board
2. **Double Bar** - IR sensor and sender arrays will be placed both vertically and horizontally

And the following variants are possible for further work:

1. **Double Bar** *(4 lines)* - IR sensor and sender arrays opposite to each other (note: size limits due to frame structure), and arranged in a square shape.[4]
2. **Single Bar frequency assisted** - Single Bar with pulses of specific frequencies sent over different senders, for a more definite signal reception.

Our engineered scripts properly digitize a discrete input such as for a 5x5 grid for 5 sensors on one side, but not for a relatively continuous input such as a 200x200 resolution for a 5+5 sensor array. However, cleanliness of the obtained values suggests that this might be possible in further work.

A key point of the system was to be as cost-effective as possible.

For further discussion, we will talk about the Single Bar design, it gave the best results and has scope for improvement.

4. the structure was ready but complicated and difficult to debug, the circuit wasn't functional. The use of Hookup Wires would have simplified this variant significantly.

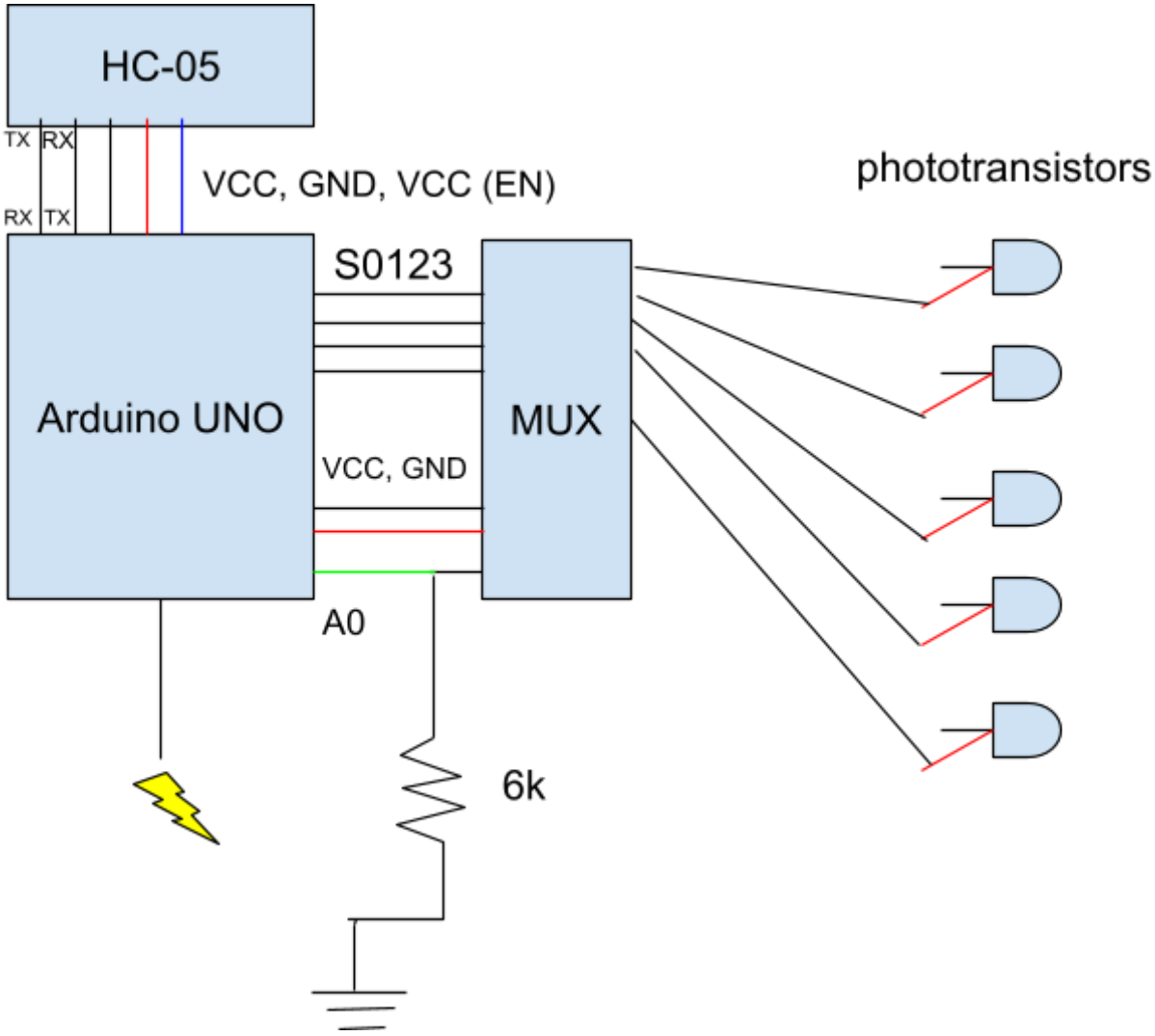
III. COST

The following components were involved in the Single Bar design:
1x Arduino UNO - **400 Rs.**
1x CD74HC4067 (16-to-1 Analog Multiplexer) - **70 Rs.**
5x IR LED - Phototransistor Pairs - **14 Rs. per pair**, total **70 Rs**, scalable for accuracy up to 16 pairs.
1x 10Ω Resistor, **1x** 6kΩ Resistor, **40x** Jumper Wires, **1x** Breadboard - variable, can consider **100 Rs. upperbound (soldered)**
1x HC-05 Bluetooth **223 Rs.**

Total: ~900 Rs., when built using general purpose resources.

For our setup, we did not require the Mux for the Single Bar design (we used 5 Pairs). And we may also do with lesser functionality than that of the UNO - and use a Nano (which also uses an ATMega328).
With these changes, we save an additional **170 Rs.**, cost coming down to **~700 Rs.**

IV. CIRCUIT



The sensing circuit is as shown. The sending LEDs can be powered externally as well for our design, since they are kept on the whole time, so not depicted here.

Note that the phototransistors have their short leg (black) (-ve end) connected to VCC.

V. BOARD LOGIC

The board simply reports the values read from A0 via Serial. There are a few quirks involved.

The Arduino UNO only has support for reading 6 analog pins, while we planned for 12 initially. So we used a digitally controlled analog Mux to give turns to each pin, since the delay for a single read was not much. The board runs through these pins, sensing each and returning the value eventually.

For the actual “sensing”, we perform a few analogReads and consider the last one, since the phototransistor takes a while to stabilize right after the path from it to GND through 10k is activated. This is the only processing done on the board. The rest of the computation is done on the host.

We have not experimented with a Serial rate higher than 9600 bps. The current latency seems more than sufficient practically for sensing.

VI. SCRIPT LOGIC

The host connects to the board via Bluetooth, and communicates with the created serial port. Initially, calibration is needed to find the ambient (surface + atmosphere) IR value, and to find the maximum and minimum achieved values - this is done by prompting the user for placing an obstacle in front of the sensors, and clearing the area. Minimum includes ambient so we do not actually use these values currently.

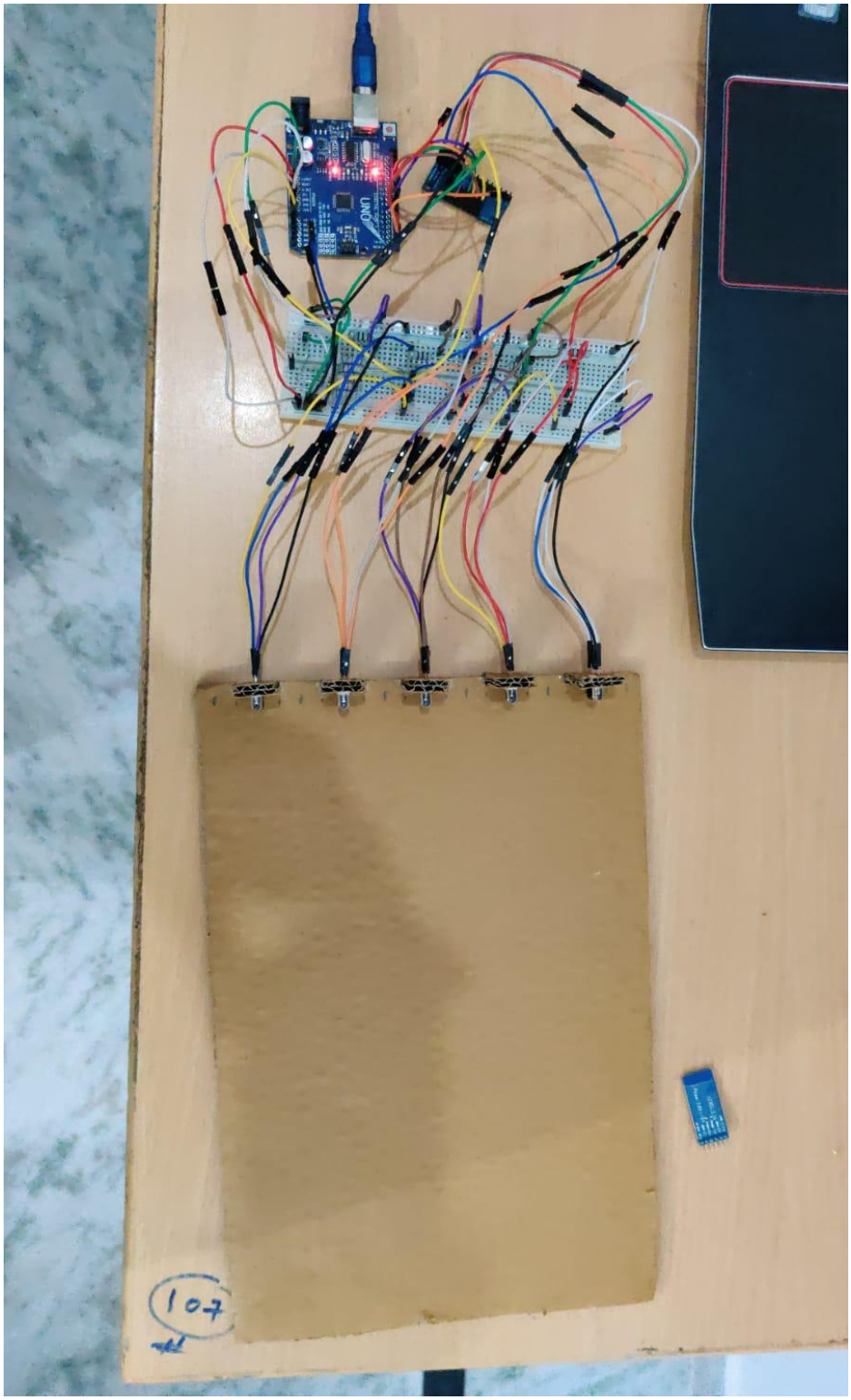
A number of reads are performed (100), and the result averaged. The sensed values here on out are normalized, for each sensor, using (current - minimum) / (maximum - minimum).

In this scheme, if the normalized value is negative, it indicates that the particular sensor is either sensing something outside the sensing area, or it is reading an ambient value.

Now, for converting the sensed values into the touch coordinate, we perform normalization as specified, and then use an intuitively fit curved to allot it into one of N discrete buckets. This value is plotted. For vertical coordinate, the maximum activated sensor is used.

Finally, we plot this value. Plotting is done using `matplotlib`.

VII. PHYSICAL CIRCUIT

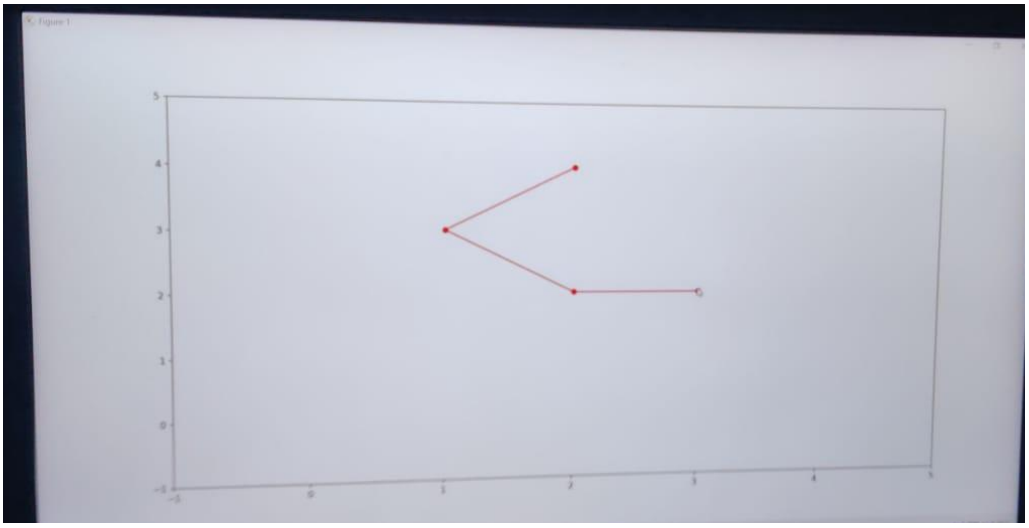


VIII. PLOTS OBTAINED

The following is a semi-processed (no `arr2touch`, `normalization`) plot of values received from the board as a finger is slid from the first sensor in the line to the last, at a fixed distance away at uniform velocity.

The dips in the sensor values can clearly be seen, with easily cleanable noise. The sensors are based at different values and some have deeper dips, but this can easily be fixed after normalization on the host.

The following is a final plot generated by our script, showing touch locations.



IX. SCOPE FOR FUTURE WORK

1. As mentioned earlier, the values are clean enough for analog sensing to be possible even with a discrete number of sensors on an axis. This is the primary area for improvement.

2. Vertical calibration can be performed to improve the intent detection for a touch.