Icescape

- Technical report -

HAW HAMBURG SS 2019

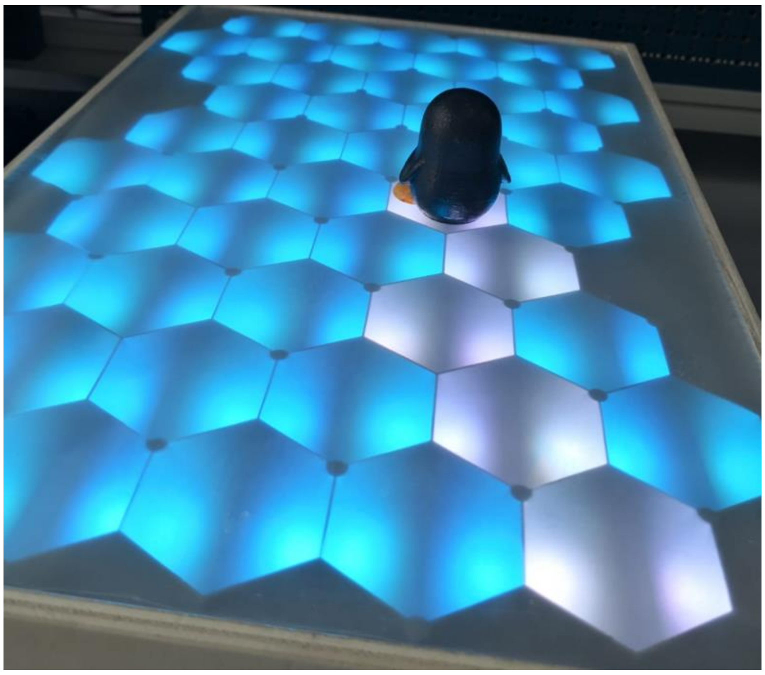
Laura Trivedi – 2260076
Laura Schlisio – 2331996
Vanessa Reuwsaat – 2259295
Simon Hoyos Cadavid – 2319104

Git Repository: *https://github.com/hoyoscadavids/icescape.git*

Introduction

The creation of projects is a natural occurrence during the time at the University. During the Mobile and IT systems lecture in the Hamburger University of Applied Sciences the students were task with the creation of an integrated system between an Arduino and a Mobile App. Following the instructions that the app should communicate with the Arduino via Bluetooth in order to control or display some aspect of the final project, it was thought by the creators of Icescape to create some sort of game, that could be enjoyed by everyone.

The main idea of Icescape is for the player to remember a path in order to get to the other side! One plays as a penguin who doesn't know how to swim and must walk through a path made of ice without falling into the water. The path is only visible for the first five seconds before everything is back to being blue, because of this, the player has to remember it and successfully get to the end. Once a wrong step has been made, the round is over, making one have to start over from the beginning, but fear not! one has as many attempts as needed.


(Picture No. 1: Icescape)

One question that might arise then is how does the game implement both the mobile App and the Arduino, which can be answered by the following statement: The Arduino is in charge of controlling the physical aspects of the game (The LEDs for example), whereas the App takes the role of the game master, allowing the manipulation of where and how the road is going to be.

Project development

Step by-Step building manual

Step 1:

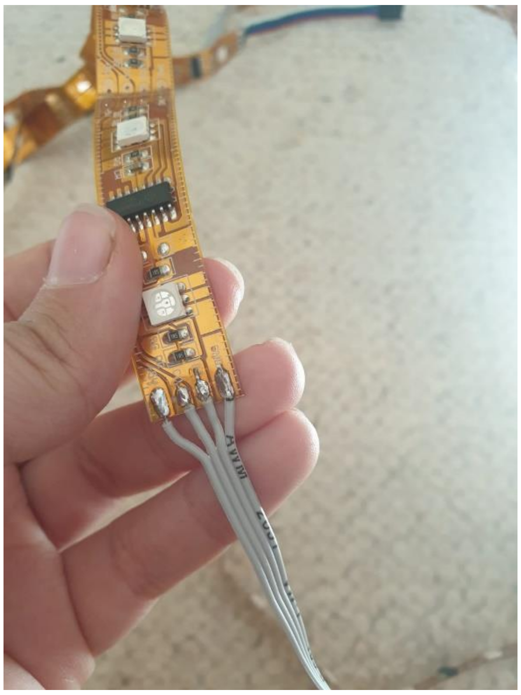
As the first step to build the game it was needed to find a shape for the "body" of the game board. It was decided to use a honeycomb structure to represent the ice floes. The structure was found within simple gravel combs for the garden.

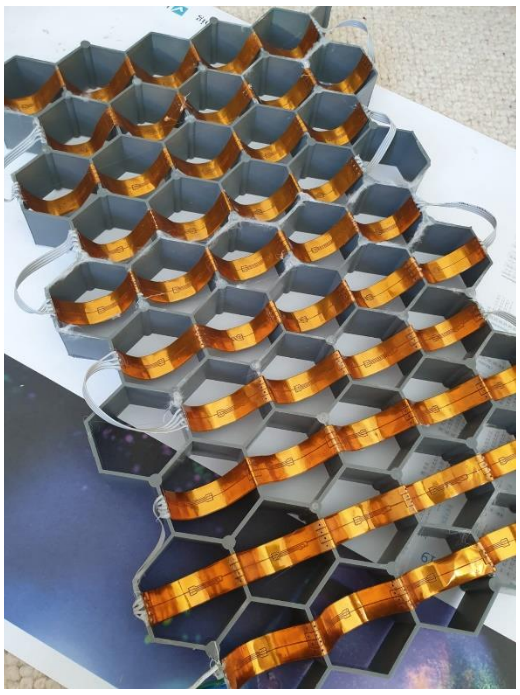
A wooden corpus was then built and an acrylic glass was places on top the honeycombs.

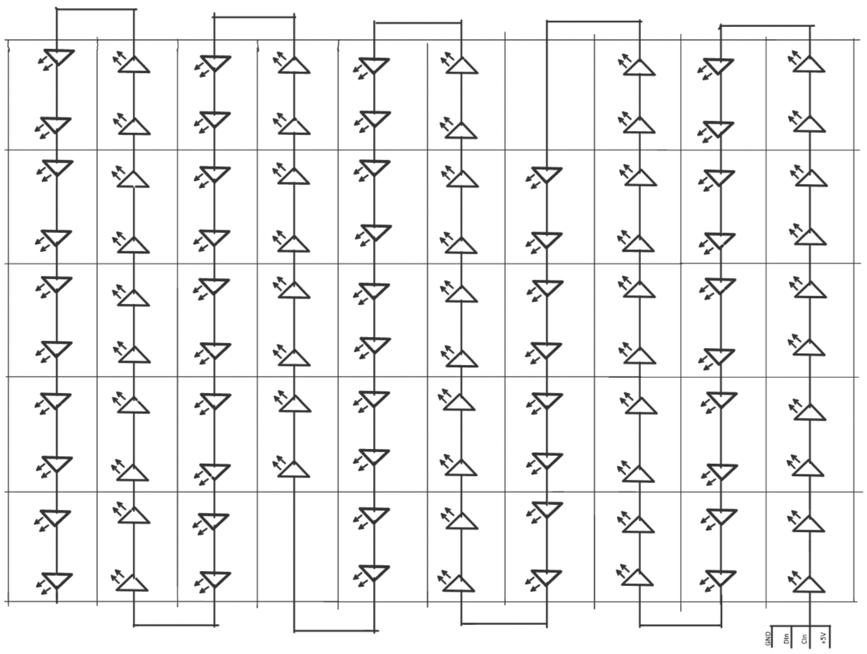
Step 2:

In order to light up the 50 ice floes in different colors we chose to use a LED-strip (LPD8806S, *see technical components*). Since the strip only contained 96 LED's (two LED's per ice floe), it was decided to minimize the game board to 48 fields.

To integrate the 48 LED pairs into the approximately 5x10 shaped board the strip was cut into chains of 5 LED pairs each (while having two 4 pair chains). The strip was then soldered from one row together with the chain of the next row via wires.


(Picture No. 2: Soldered chains)


(Picture No. 3: All connected LED chains)

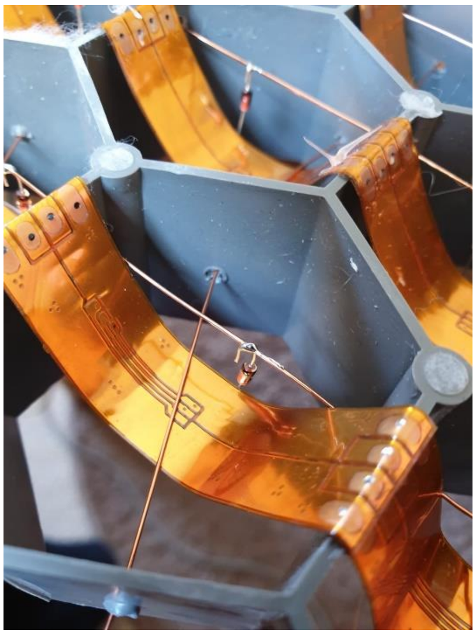


(Picture No. 4: Circuit diagram of the LED chain)

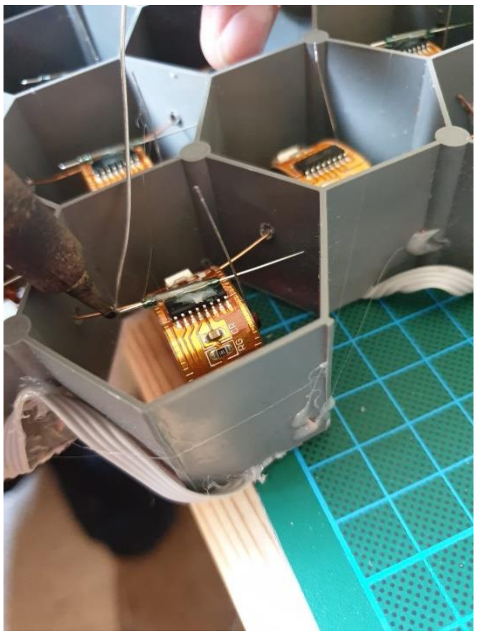
Step 3:

As the next step two wires were put through the combs as a preparation for the reed matrix. The reed matrix consists of 48 reed switches and a diode (*see technical components*), for each floe.

Both components were soldered together, the "reed side" was connected with the vertical wires and the "diode side" with the horizontal wires.



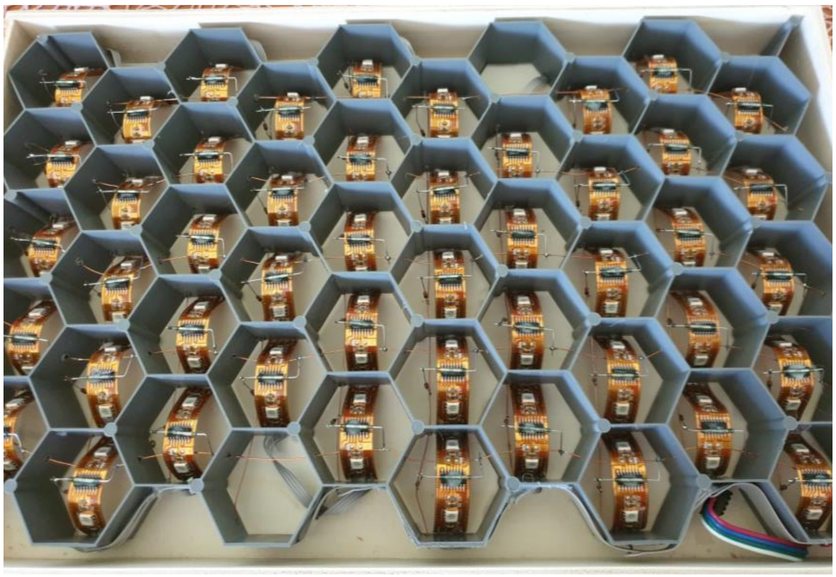
(Picture No. 5: Connection of the diode side)



(Picture No. 6: Connection of the reed side)

Step 4:

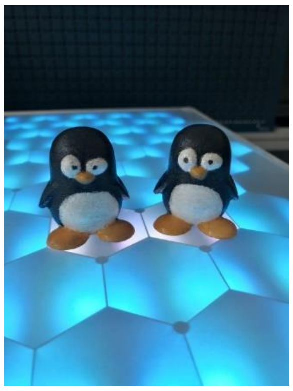
The 5 and 10 wires of the matrix were brought to the point where the 4 wires of the LED-Strip were (Clock, Data, VCC and GND) and connected the 15 wires of the matrix, the Clock, GND and Data Pin of the LED-Strip with the Arduino mega. (The VCC was fed separately.)



(Picture No. 7: Finished technical structure)

Step 5:

To trigger the reed sensors, a magnet was put into the body of the game characters. These penguins were previously printed with an 3D-printer from a template which was found in the internet.


(Picture No. 8: Game characters)

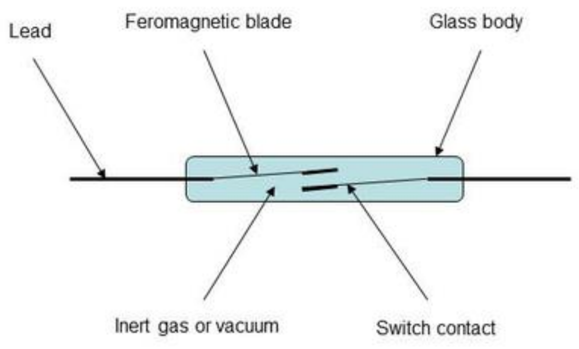
Technical components

LED-Strip (LPD8806S):

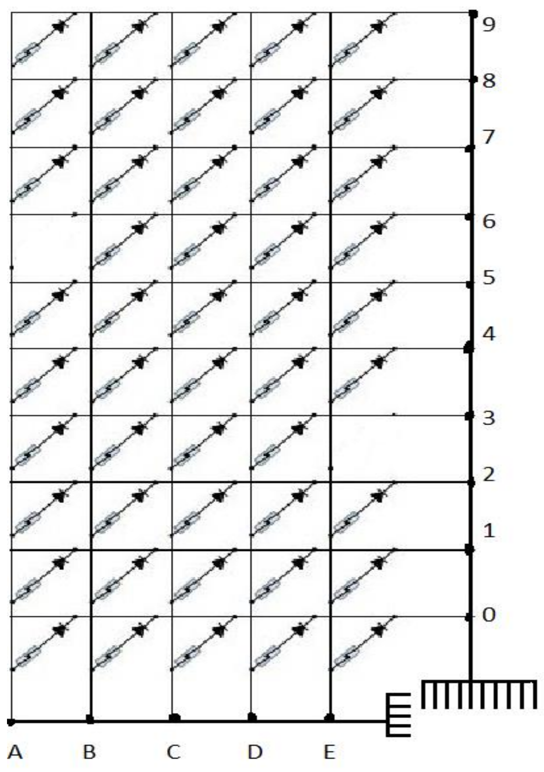
LPD8806S uses CMOS processors and has 4 pins. GND, CLOCK, DATA and VCC. The strip needs a voltage of about 5 volts to operate which was needed to feed in separately. The CLK (clock) pin is used to „clock“ data out of the DATA pin. During one cycle of the clock pin, one bit of the data information is transferred to the DATA pin.

Reed sensors:

The reed switch consists of a ferromagnetic “blade” and a switch contact within a glass body. *(See picture No…)*. Once a magnet enters the magnetic field of the sensor the switch closes and the closed sensor triggers an impulse into the Arduino. A Diode is soldered to the switch to prevent the current to flow into the wrong direction. To minimize the number of wires to 15 we arranged the reed sensors and the diodes in a matrix.



(Picture No. 9: Reed Switch)



(Picture No. 10: Circuit diagram of the Reed Matrix)

Arduino Programming

For the most part, the Arduino was in charge of controlling the physical components of the project Icescape:

- LEDs: The whole LED strip was made to contact with the Arduino in order to change color, with the function colorWipe. The idea behind it was that depending on the field where the player stepped it should either be colored white, when correct, or the whole board should blink red and restart the game when wrong. Another feature was of course that if the player got to the other side, the whole will also blink green several times.
  It might be important to clarify that the wave effect that the game has was made on purpose, by refreshing the LED strip each time a single panel was changed; with the exception of when it was colored white.
  The LEDs were represented as a single dimensional array.
- Reed Sensors: The reed sensors sent a signal several times per seconds, represented as a two-dimensional array of ones and zeroes. Whenever a zero was detected, meant that the player was stepping on that field. With this in mind a little bit of trickery had to be implemented in order to pair it with the LED array.
  First of all, the array had to be transformed into a matrix, which also had every even column reversed. This was done because of the orientation of the LED strip, going up and down in zigzag.

The second correction that had to be done was because of the fact that the number of fields in the matrix was 50, whereas 48 in the LED array. As seen in some pictures beforehand two fields were taken off the end matrix, which called for the usage of two offset values (1 and 2) after the corresponding missing fields.

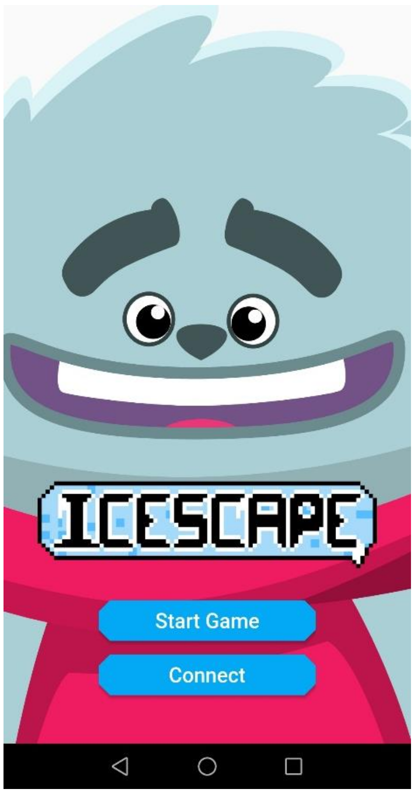
- Bluetooth: The third and final task of the Arduino was to communicate with the mobile App via the HC10 module. This was a really complicated thing to accomplish because of several complications that will be explained in the programming of the application, but it should be stilled noted what exactly the Arduino do with the data coming from the Bluetooth. The aforementioned point can be explained with the Arduino taking in several arrays of binary data and packing it in a 48-bit long array, representing with ones the correct fields (or the path to be walked) and with zeroes the rest.

App programming

As stated before the Arduino had to be connected via Bluetooth with a mobile application, which could use any programming language the students picked. For this purpose, it was then decided upon to use Flutter, an SDK developed by google for mobile development utilizing the programming language Dart.

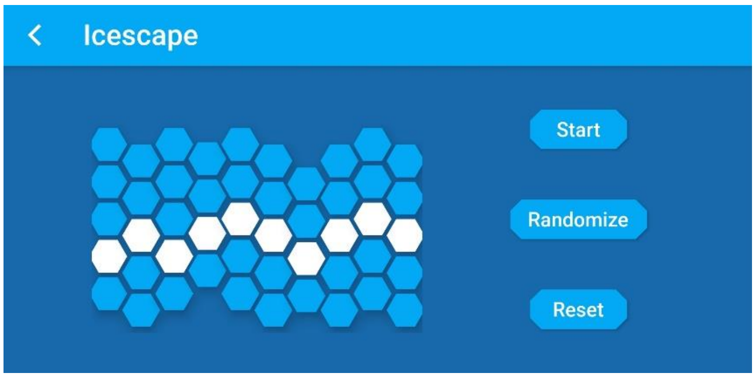
As the code will be annexed with the project, not much has to be explained concerning how it is structured. But as a basis it can be clarify that it consists of two main screens being a start page and a game page, where the user can connect to the Arduino and create the path to start the game respectively.

For the start page, not much is to be said apart from it consisting of two buttons for either starting the game or connecting to the Arduino, accompanied by a self-made logo and an open-source Flare animation of a bear like creature. The connection button can only be used if the user is not yet connected to the Arduino, which will also, when tapped, prompt the user with the information needed (either a successful or not connection). As a last note, the Arduino's Bluetooth must be named "icescape" because the need of making an extra page in order to select the Arduino to connect with was deemed unneeded and not searching through UUID allowed to change the Arduino or module to another one with the same name without needing to change the code.



(Picture No. 11: App's Start Page)

The main game page involved utilizing a series of 48 hexagon shaped buttons in order to create a grid similar to the one shown by the hardware. This part of the application was built utilizing the row and column layout provided by Flutter, where a function is in charge of building the same columns several times in order to create a 5X10 grid.



(Picture No. 12: App's Game Page)

Next to the previously mentioned layout three buttons were placed. Starting from the button the reset button cleared the whole hexagon array and the randomize button created a really easy random path, following an algorithm that looked what the next two free cells were. On top of these two buttons the Start was placed, which was in charge of sending the information to the Arduino via Bluetooth.

The application was locked on landscape mode for this page in order to keep the given layout.

Bluetooth explanations and complications

One of the most daring aspects of the programming side of the application was the Bluetooth module of the Arduino. Not much difficulty was found when connecting it to the Arduino persé, but instead the usage of Bluetooth in a mobile app is not as easy as one would think.

First of all, one had to get into the understanding of how the BLE protocol works. A Bluetooth instance with a service must be found, having a UUID, which has a characteristic that allows either write or write without response, and when this is found, a message can then be sent. When all of this is identified, the connection can be stablished, and messages can be sent to the Arduino. It can be noted that even if this is a short explanation of how the BLE works, it took a long time while developing the App in order to achieve the desired outcome.

Another important point to clarify is how the data is sent, because the HC10 module can only send around 20 bytes at a time. Since the data that had to be transmitted was around 48 bytes (in actuality it could be 48 bits, but the data had to be sent in character form, being each character one byte), it had to be split in the following "protocol":

- A number between 0 and 2 explaining the place of the string in the array.
- A plus character: "+", signifying that the data came next.
- 16 Characters of data (Either ones or zeroes).
- A minus character: "-", signifying the end of the string.

Giving the following form 0+0000000000000000-. These strings represented the data array divided in three subarrays with a length of 16, which was then reformed into one array in the Arduino.

Sources:

https://cdn-shop.adafruit.com/datasheets/lpd8806+english.pdf

(Last visit: 07.12.2019)

https://en.wikipedia.org/wiki/Reed_switch

(Last visit: 07.11.2019)

https://www.reichelt.de/reicheltpedia/index.php/Reedschalter

(Last visit: 07.12.2019)

https://sites.google.com/site/bergersprojects/reedcb

(Last visit: 07.13.2019)

https://forum.arduino.cc/index.php?topic=253195.0

(Last visit: 07.12.2019)

Picture Sources:

Picture No. 1-8, 11-12: self-made

Picture No. 9: https://www.reichelt.de/reicheltpedia/index.php/Reedschalter

Picture No. 10: https://sites.google.com/site/bergersprojects/reedcb/matrix