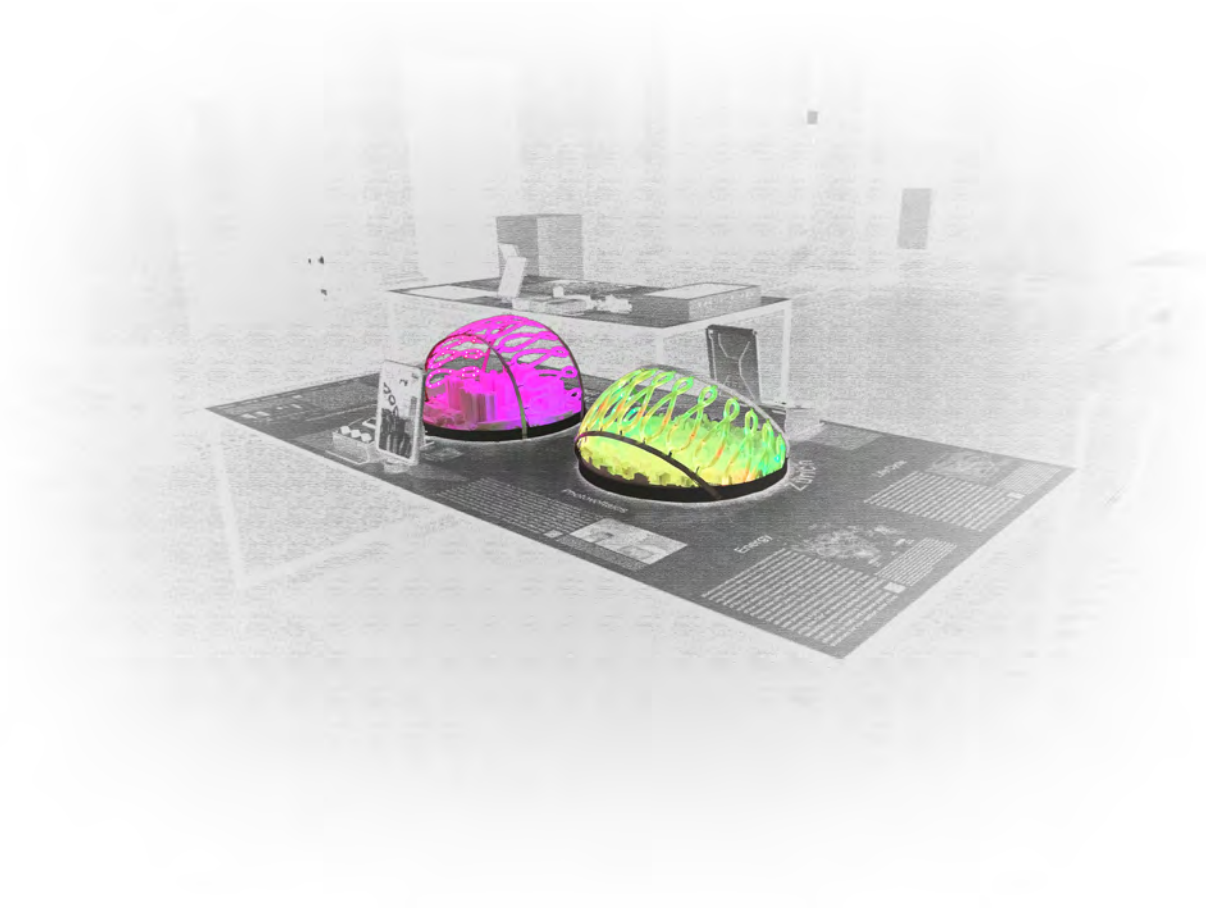
Master in Integrated Building Systems MIBS

ETH zürich

Semester Project

Interactive Data Visualisation on a 3D-printed Sun Path Model



Rino Sogno

E-mail: sognor@ethz.ch

Zurich

October 20, 2023

Contents

1 Introduction **3**
- 1.1 Framework . . . 3
- 1.2 Targets and Milestones . . . 3

2 Thoughts about the real sun path **4**
- 2.1 The locations . . . 4
- 2.2 The sun path in reality . . . 4
- 2.3 The analemmas . . . 4

3 Conceptualizing the sun path in a visualisation **5**

4 Prototyping of different approaches **6**
- 4.1 The mechanical approach . . . 6
- 4.2 The light approach . . . 9

5 The first user feedback **13**

6 Details of the final design **14**
- 6.1 The 3D printed city model . . . 15
- 6.2 The base . . . 16
- 6.3 The sunpath and the analemmas . . . 19
- 6.4 The user experience . . . 20
- 6.5 The Electronics and the Code . . . 23

7 The final result **27**

References **31**

Annex **32**

Acknowledgement

Thank you to the Chair of Architecture and Building systems, who hosted this semester project. Especially to my supervisor Christoph Waibel for his support and the good collaboration.

A special appreciation to my friends, who supported me in the last phase of the model assembly. Thank you very much for all the many hours spent on the very repetitive tasks of soldering, gluing, casing LEDs and sorting cables.

1 Introduction

1.1 Framework

As part of the international Future Cities Laboratories (FCL) conference conducted in October 2023, the FCL research program hosted a public exhibition in the main hall of the ETH Zurich main building. The goal of the FCL is to help shaping sustainable cities and settlement systems through science, by design, in place, and over time.

Within FCL, the "Powering the City" (POW) module, an international consortium of researchers from Singapore, Sweden, Switzerland, and the USA, were represented with an exhibition stand to showcase their ongoing research. The objective of the module is to investigate the deployment of building-integrated photovoltaic (BIPV) in urban environments, with themes revolving around it, such as socioeconomic, decentralized renewable energy, Vehicle-to-Grid, life cycle emissions, and architectural design.

For this POW exhibition stand, we aimed to develop an interactive installation, involving electronic (and potentially mechanic) artefacts displaying the sun paths of Singapore and Zurich, as well as interactive data visualization techniques involving Augmented Reality (AR). The stand should combine information of the ongoing research, physical objects of BIPV panel prototypes, the interactive artefacts, as well as (touch-) screens to interact with the artefacts.

1.2 Targets and Milestones

Within the aforementioned framework, this semester project was focused on the conceptualization, detailing and fabrication of the interactive artefact visualising the sun paths for the two studied locations (Zurich and Singapore). The main goal was to decide upon the mode of visualisation (mechanical, light sources, etc.), the information visualisation control using an Arduino board, and to determine the design of the artefact as an art object. Once a final concept was derived, construction plans and details were produced and the artefact was manufactured The following main milestones were recognised:

1. Prototyping of the representation of the city scape (potentially 3D printed).
2. Exploration of several technical options on how to visualize the sun path. This involves a static element showing the bounds of the sun paths (solstices) and some chosen sun paths, which could potentially be made from metal or wood. On the other hand, there is an interactive element (eg. using an LED-Mesh, using individual LEDS, using a mechanical element, a moving belt, or similar) to dynamically visualize the sun position or other information for different scenarios.
3. This exploration was followed by a prototyping and ideation of the possible realizations of the most promising concepts on a small scale to prove its feasibility and visualization potential.
4. This was combined with an exploration of the potential to show further information beyond the sun path (eg. PV penetration, grid emissions, temperature of the hour in a TMY, etc.) using the interactive representation of the sun. This involved programming a board (such as an Arduino) as a controller using the data given.
5. Potentially creating a connection to an interface allowing the visitor to play (potentially involving "mini games") or interact with the information shown and decide what he wants to see. This topic has an overlap with another ongoing semester project developing the AR setup and depends on the data given from the simulations.
6. Once a final concept was derived, production plans and details were drawn. If needed, these were discussed with the respective ETH workshops or third party service providers, such that it was possible to produce the final product in time at its actual scale for the exhibition in October.

2 Thoughts about the real sun path

2.1 The locations

Both locations, Zurich and Singapore, lie on the northern hemisphere. Therefore, in both cases, the sun will come from the southern direction (cf. *Figure* 1). However, due to the close proximity of Singapore to the equator, the sun angle will be almost perpendicular to the ground. Thus, Singapore will show very little seasonal variation, as the sun angle does not change significantly over the seasons. In contrast, in Zurich, the sun angle is low and thus, seasonal variability is expected to be seen.

Figure 1: The two studied Locations on the world map

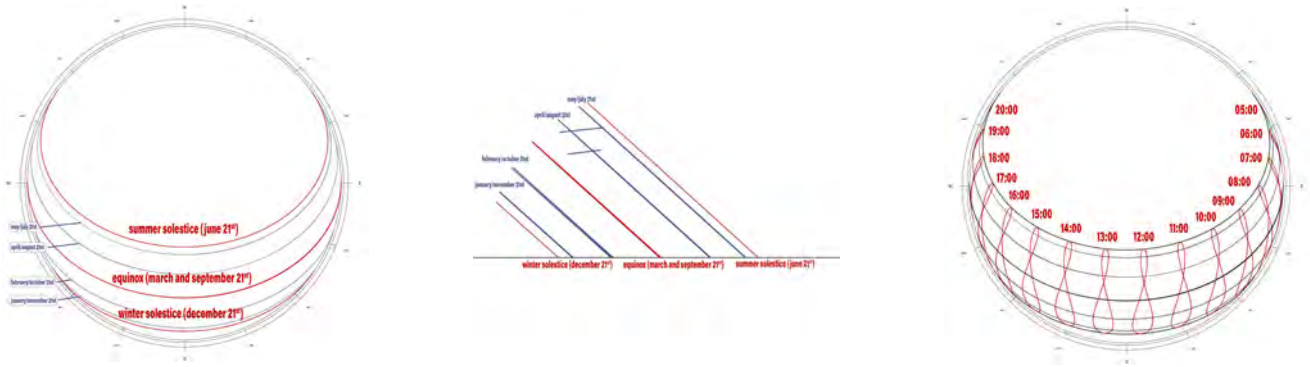
2.2 The sun path in reality

Taking a detailed look at the sun paths for the two locations, one can see, that the possible sun positions lie in an area bounded by the horizon at the sunrise/sunset positions and the paths which the sun takes at the summer/winter solstice respectively. Every possible sun position must lie within this area.

In general there are two interesting aspects to the movement of the sun. Over the course of a day the sun moves from the east to the west (for the northern hemisphere) on its daily path. However, the path is not the same for all days. Over the course of the year, the sun path moves. This movement goes from the longest path at the summer solstice to the shortest path at the winter solstice. In the middle of this movement, one can find the equinoxes (cf. *Figure* 10).[1]

2.3 The analemmas

An often used concept in thinking about the sun's path are the analemmas. These are the shapes created by looking up onto the sky at the same hour every day of the year and marking the position of the sun. In doing so, one will realise, that the sun's positions at the same hour over one year moves along a shape looking similar to a figure eight (cf. *Figure* 2). This shape is caused by the eccentricity of the earths orbit (different velocities at different positions) and the fact that the plane on which the earth rotates around the sun is tilted compared to the plane in which the earths rotation takes place (Eclptic).[2]



Top view Side view Analemmas

Figure 2: Visualisation of the change of sun path over the seasons

[1] Daniel V. Schroeder, 2011.
[2] Stanford Solar Center, 2022.

3 Conceptualizing the sun path in a visualisation

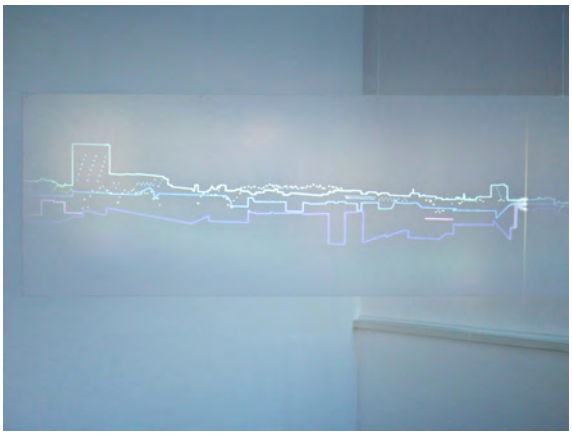
Starting from the aforementioned explanation, there are two major ways to go in terms of interactive sun path visualisation: One possibility is to try to mimic the movement of the sun using actuators and move a light source in a continues movement across the sun path area. Lets call this the "mechanical approach" (cf. *Figure* 3). Another possibility is to choose a discrete set of positions on the sun path area (grid) and visualise this set as a mesh of individual light sources. Lets call this the "light approach". The use of LED lights is common in interactive exhibition setups (cf. *Figure* 4). In the following project both approaches where explored and the "light approach" was chosen for realisation in the end.

HPD Model 126 Heliodon [3]

Heliodon for daylight analysis [4]

Figure 3: Different mechanical setups used in the visualisation and exploration of the sun path

The dimensions of the sun's path are a many fold bigger than we humans are. As a result, we are always looking up to the sun or the sun path and the sun shines down on us. However, looking at the planned exhibition model, the models dimension is way below our human scale. Meaning the perspective shifts from a looking up to the sun to a looking down at the sun path model and the model of the city. We are no longer immersed into the cityscape and the sun system, but suddenly in a floating outside position. This means for the visualisation, that the light source(s) mimicking the sun not only need to shine onto the model of the city (the usual perception of the sun shining down onto the earth), but also needs to shine away from the model towards us, the viewer, such that we can localise the position of the sun looking from above.



Sensing the City - interactive sensor data visualisation [5]

community-made, interactive town map [6]

Figure 4: Visual exhibition setups using lights do display information

[3] (Laboratory of Integrated Performance in Design, 2023)
[4] (Kyle Konis, 2020)
[5] (Andreas Muxel, 2014)
[6] (Arduino Team, 2016)

4 Prototyping of different approaches

Based on the aforementioned thoughts there are two directions to explore. A moving mechanic or a grid of lights.

4.1 The mechanical approach

Looking at the area in which the sun moves over the course of the year, it can be seen, that we are dealing with at least two dimensional movements. Thus, our mechanical system will need at least two actuators. Thinking of this surface as a two dimensional space, one can choose the base vectors (and thus, the two directions the actuators act in) in a variety of ways. However, there are two choices of coordinate system, where the directions potentially have an intuitive meaning to the viewer. These approaches are described and explored below.

4.1.1 Horizontal Coordinate System Model

The horizontal coordinate system is often used to describe the position of a star (eg. the sun). To do so, the position is split into an Azimuth and an Altitude Angle. Therefore, using this model to create a mechanic to move a light source simulating the sun, means having a rotational movement and a movement along a bow in vertical direction (cf. *Figure* 5)
A prototype was built exploring the possibilities which such a mechanic setup has. This setup would need two stepper motors and a turn table around the model carrying a bowed rod, along which the light source (sun) can move up and down (cf. *Figure* 6). The downside of this coordinate system is, that to trace the path the sun takes over a day, a simultaneous movement of both motors would be necessary, which makes the programming of the movement complex. Furthermore this choice of coordinate system has a disadvantage in its appearance, as the visitor sees the two axes, but not the trace of the actual sun path. From an educational standpoint it would be more interesting to make the sun path visible for the viewer. To do so, an additional installation (eg. metal wire mesh) would be needed.

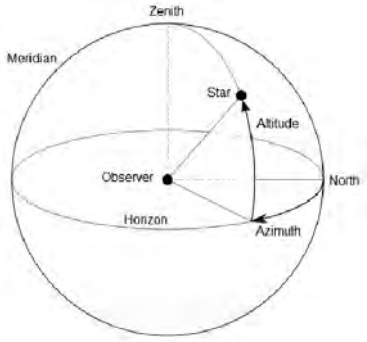
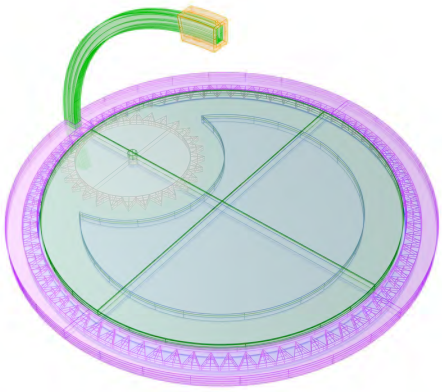
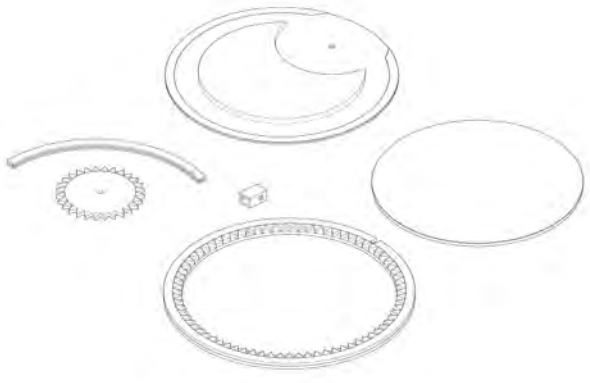


Figure 5: The horizontal coordinate system [7]

Figure 6: Fully functional prototype using a horizontal coordinate system



Perspective of the mechanic



Individual Parts of the mechanic

Figure 7: CAD plans of the horizontal coordinate system prototype

[7] (Szücs-Csillik, Iharka and Poputa, Dora, 2015)

4.1.2 "Sun path Coordinate System" Model

As seen in *Section* 4.1.1, it would be beneficial from an educational standpoint to have the sun path over a day clearly visible to the visitor. To do so, the movement of the sun over a day can be split differently. Namely into a offset of the path between the two solstices and a rotational movement along the path (cf. *Figure* 8). Let us call this reference system the "sun path coordinate system".

However, this brings the disadvantage, that it leads to a presence of the path below the model surface for some positions, as the winter season has shorter paths than the summer season. This leads to more space needed below the model. To prevent that, in a first attempt a prototype was created working with a bendable circle (here nylon wire, in a larger scale probably spring steel rod) representing the sun path, which was fixed on the underside and shortened or lengthened using a slider (cf. *Figure* 9). The slider could potentially be moved back and forth using a gear and rack system. (cf. implementation in *Figure* 14). This system could be scaled up using a spring steel rod to recreate the bending behaviour of the nylon wire. However such a slider mechanism, which changes the exposed length of the system, makes the rotational movement of the path very difficult to control and implement because of the bending. Therefore a system without the bending of the sun path circle was explored.

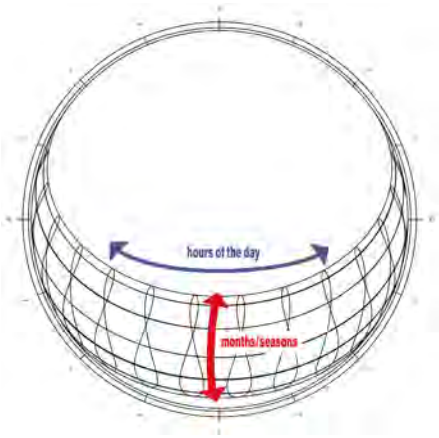
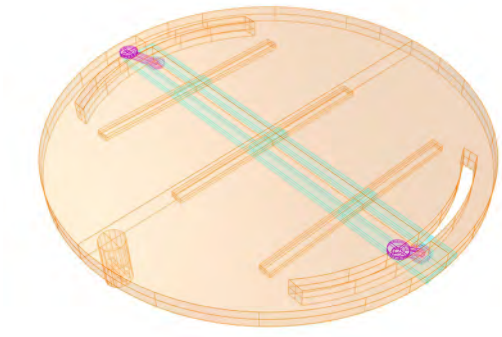


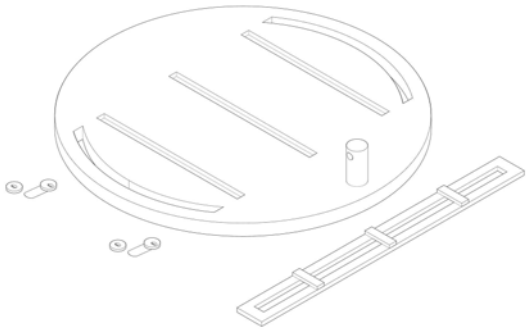
Figure 8: The "sun path coordinate system"

Figure 9: Fully functional prototype using an sun path which can be offset

Figure 10: CAD plans of the horizontal coordinate system prototype



Perspective of the mechanic



Individual Parts of the mechanic

Figure 11: CAD plans of the horizontal coordinate system prototype

[7] (Szücs-Csillik, Iharka and Poputa, Dora, 2015)

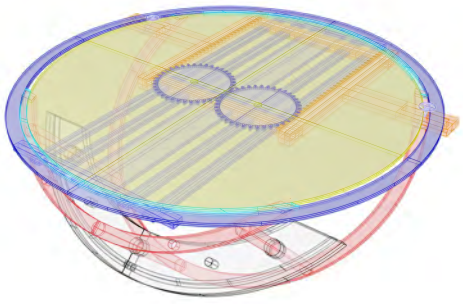
4.1.3 "Sun path Coordinate" - a second attempt

Starting from this challenge, another prototype was developed. In this version, the ring is reduced to the part of the path, which is fully exposed in the most extreme position (june 21^{st}). This allows to reduce the space which is needed underneath the model compared to a version, where the path is implemented as a full circle. Below the model a stepper motor is placed using a threaded rod to move the path "up and down" (cf. *Figure* 13). To hide the space needed for this mechanical setup, a cover on top of the table would be needed. For example by angling the information texts towards the visitor and creating a plinth for the model (cf. *Figure* 12). This plinth will host and cover the mechanical assembly, such that it is not visible from the visitors point of view. The rotational movement would be realised with a wagon, which is clamped to the path and slides along the ring.

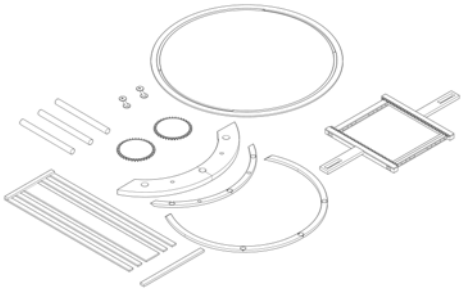
The major downside of such an assembly is, that the movement of the wagon is easily realizable but very hard to control, as the path changes its length over the seasons. Thus, there is no properly defined end position of the path and as a result no good space for an end switch to determine/reset the stepper motors position. Due to these complexities with this strategy, another strategy was chosen in the end.

Figure 12: Example setup of the table with a plinth

Figure 13: Slider setup which would be driven up and down by threaded rod and stepper motor



Perspective of the mechanic



Individual Parts of the mechanic

Figure 14: CAD plans of the horizontal coordinate system prototype

4.2 The light approach

In parallel to the development of a potential mechanical setup, experimentations were made with using a LED light grid to display color coded information on the sun path. This poses three main questions. How should the grid of LEDS look like (shape, density, brightness, LED size, ...) and what does this grid have to do with the actual properties of the sun paths. Furthermore, how could a control system be implemented to manage LED color, state and brightness individually. These questions will be answered in the following sections.

4.2.1 The shape of the grid and the role of the analemmas

The installation's aesthetics were developed starting from the question of "How does the grid look like and how does it connect to the real sun path?". One of the key decisions was not to choose an equally spaced grid, but to represent the structures of the analemmas (cf. *Section* 2.3). By not having an equally spaced grid, but a grid based on the positions at the 21st of every month one is able to show and explain the characteristic shape of the analemas to the visitor. Furthermore by choosing the day of the 21st in every month ensures that the solstices as well as the equinoxes are part of the grid (namely lowest, highest and middle positons on the analemma shape).
In my opinion, it makes sense to base grid on the yearly change of position for a certain hour (analemma), as these shapes also provide a certain aesthetic expression, I personally associate with a sun path. In my opinion, this leads to a more desirable expression of the whole artefact than if realised with an equally spaced grid or the daily paths of the sun ("semicircles") as the base shape.

4.2.2 Viewing the analemmas from the top

As the model is viewed from the top by the visitor, most of the analemmas will be viewed from a top perspective. This is especially true for the Singapore case. As the LED lights have to shine down onto the model for the purpose of shadow casting, an opaque housing, such as the one used in the beginning (cf. *Figure* 15), is not a solution, as the LEDs will not be directly visible for the viewer. Therefore, the LEDs need to be made visible from the top view.

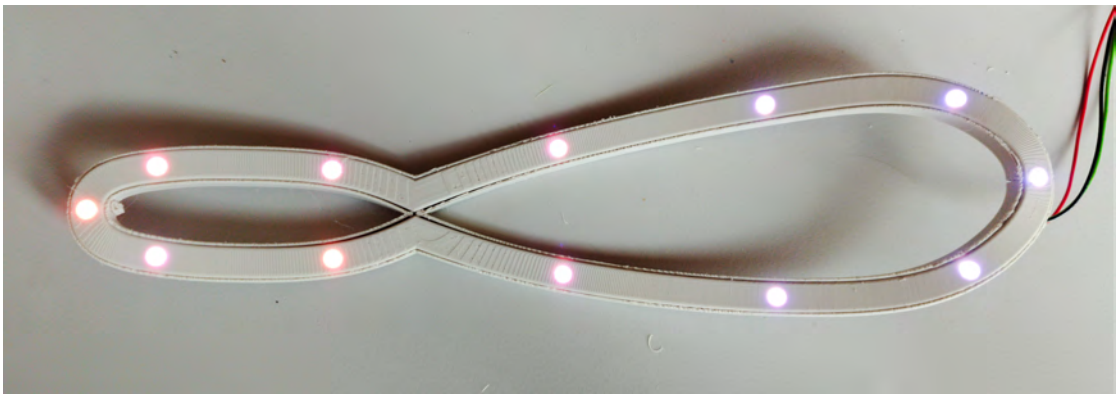


Figure 15: First analemma prototype, opaque housing

To adress this problem, a certain transparency needs to be introduced into the housing of the LEDs. In that way the viewer can determine from the top view, which LEDs are active. Two options were tested, a fully transparent analemma (base and lid transparent) and a transparent analemma (base transparent, lid opaque) with a closed back. In the first one, the LED lights are clearly visible shining through to the back. In the second one it is more a small dim glow at the side. Therefore, the fully transparent analemma was chosen (cf. *Figure* 16).

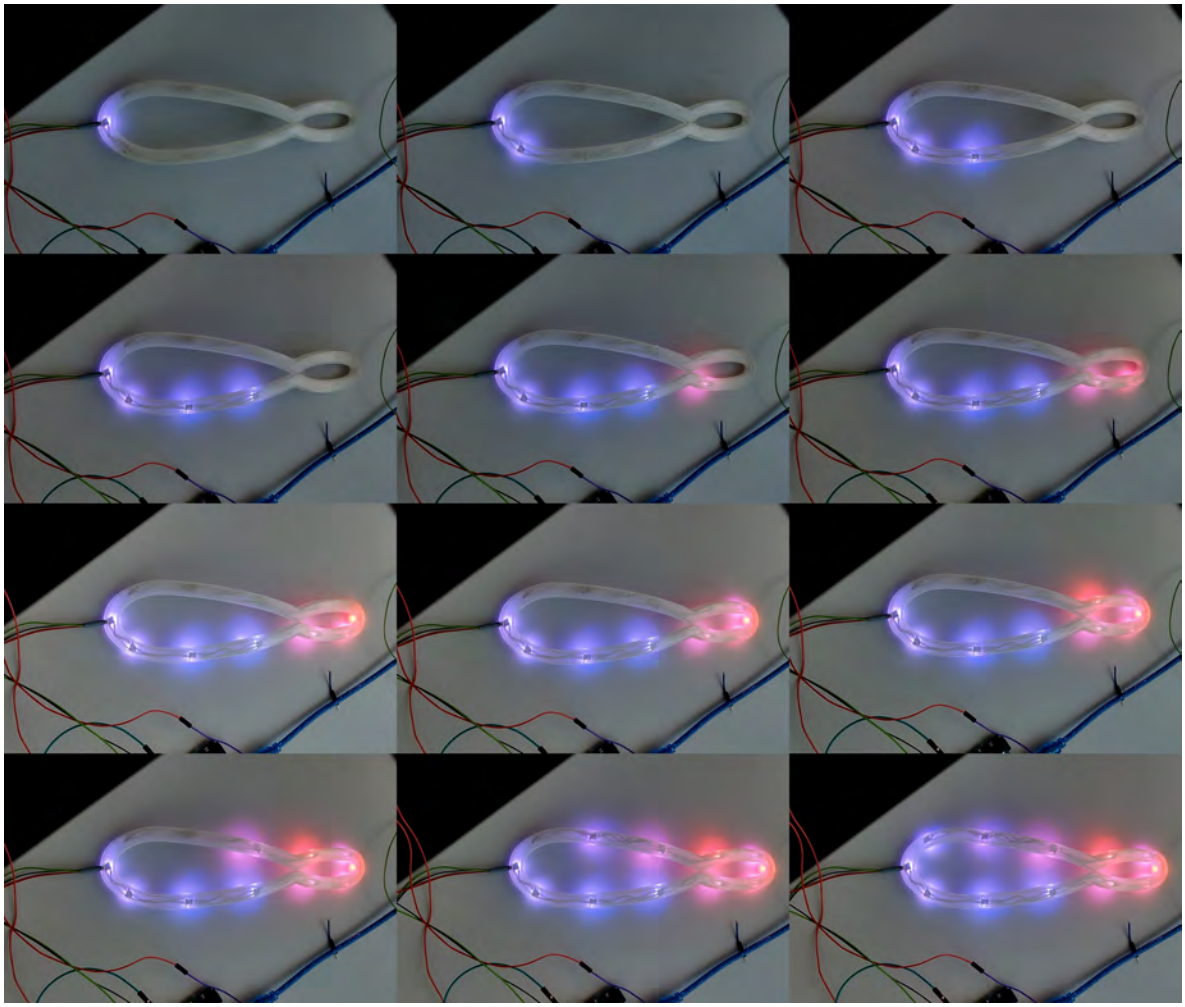
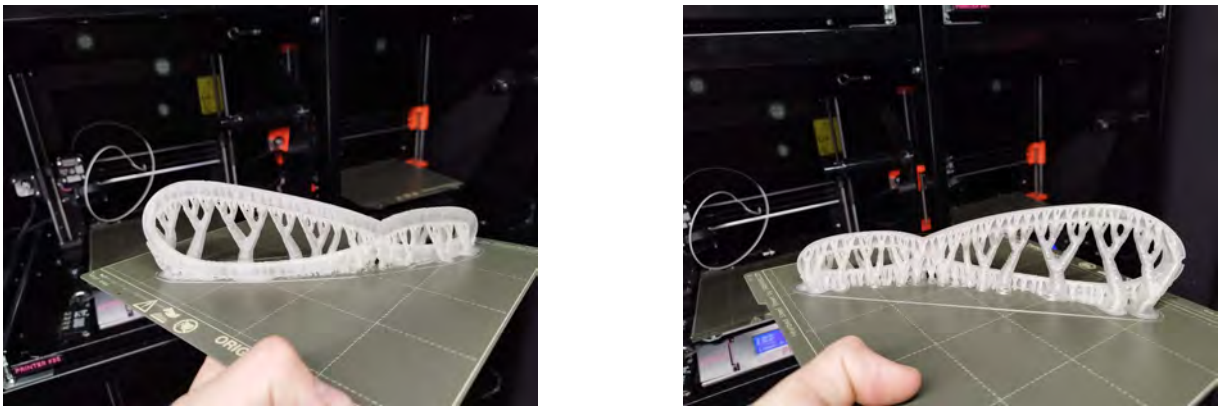


Figure 16: Glow of the LED lights through a fully transparent Analemma (base and lid)

To improve the transparency of the 3D print, the number of top and bottom layers was reduced to 2 layers, such that the printed surfaces are as thin as possible. Furthermore, by printing the analemmas standing on their side (cf. *Figure* 17), a more optimal layer orientation can be achieved, such that the transparency is further improved (cf. *Figure* 18).

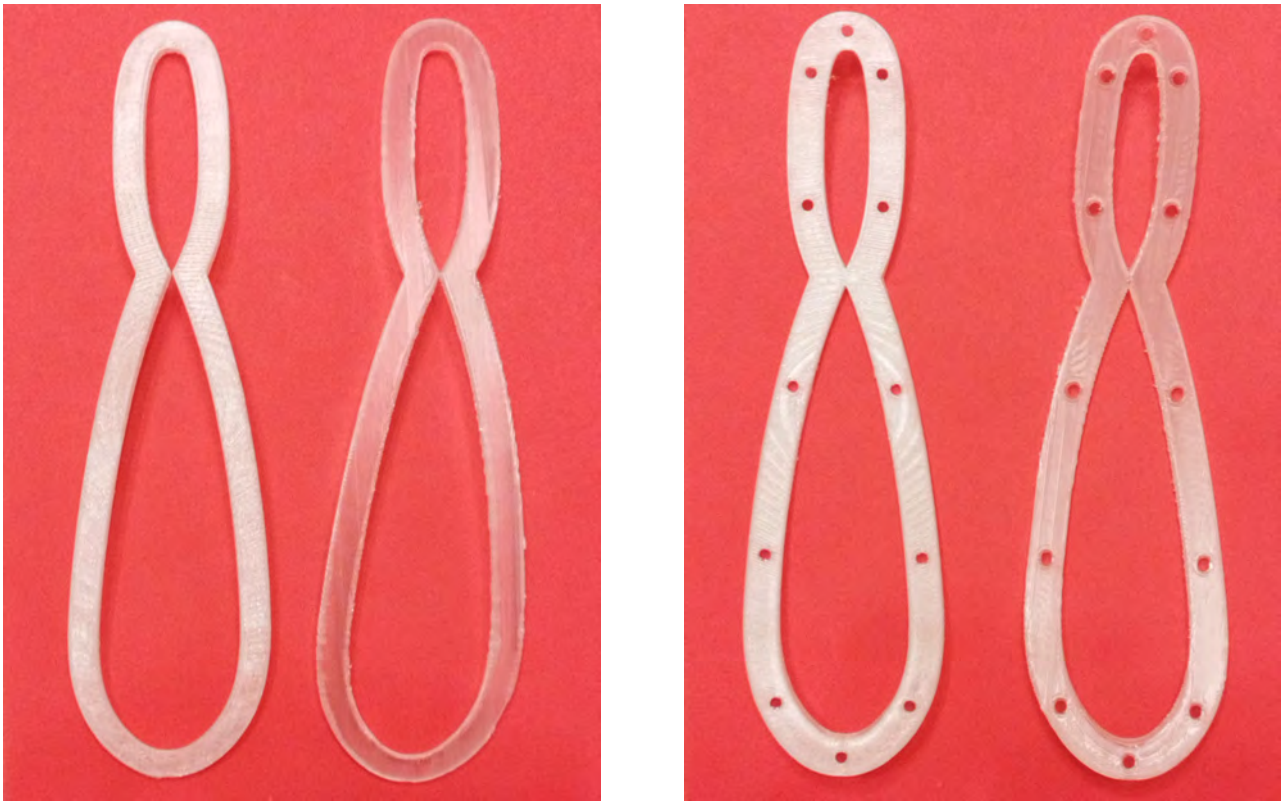


front view back view

Figure 17: 3D printing in a standing position for a more optimal layer orientation and as a result higher shell transparency

The shape of the analemma was first modelled using a simple offset of the analemma curve retrieved from the ladybug simulation tool. This offset led to an analemma shape, which was not connected in the middle (cf. *Figure* 18). However, as the analemma normally is association with the figure eight, it made sense to rework the housing and change it to an offset of the curves along a sphere. This led to a housing with a very nice looking figure eight shape (cf. *Section* 6.3).

After quite a large number of iterations the shape was finalized and the lid was in a state, where it had a perfect fit into the base and a good transparency. For pictures of the final version of the analemma see *Section* 6.3.



lid base

Figure 18: Influence of printing orientation on transparency Left: lying printing orientation Right: standing printing orientation

4.2.3 Controlling the LEDs

To properly visualize information upon the sunpath model, it is necessary to control each LEDs color, state and brightness individually and simultaneously. The chosen representation of the sun path boast approximately 140 LED lights. For that reason an approach, where the lights are steered individually (meaning that every LED would need a pin of the Arduino to be controlled), or in a Matrix (meaning that every row and every column needs a pin) is infeasible. The first solution is infeasible as the Arduino only has 54 digital pins and the second one has limitations, as not all combinations of LEDs being on and off are possible by controlling rows and columns (no independent control of the individual LEDs). Therefore a solution using a "LED chain" was chosen.

The LED chain consists of LED lights which are all wired in parallel to the power source and controlled using a data wire connecting all of the LED lights (cf. *Figure* 21). Meaning, that the control inputs and outputs of the individual LEDs are connected to a chain (cf. *Figure* 22). The Arduino is used to send a signal to the first data input. The controller on the LED "cuts off" the first 32bits of control information and forwards the rest of the signal after amplification to the second LED, which takes the originally second 32bits (now the first 32bits in the forwarded data) as control input and so on. Using this technology, the whole chain of LEDs can be controlled with only one pin of the Arduino. The product chosen in our case is from "Intelligent LED Solutions" (ILPL-K501-RGBW-SK105-01 - SMD-LED) (cf. *Figure* 19).

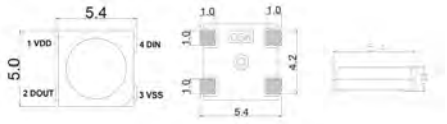


Figure 19: Chosen Product from Intelligent LED Solutions [8]

The LED's used are RGBW LEDs. This means that each of the LED units actually consists of 4 LED lights, one red, one green, one blue and one white. Thus, colors and brightness can be adjusted individually using an RGBW information of the scheme GBRW (cf. *Figure* 20). This signal is transmitted using a frequency of 800khz and a transmission delay per LED of maximal 500ns.

Figure 20: GRBW information scheme visualisation [9]

The LEDs use the SK6812 standard for the data stream. The SK6812 control data was generated using the Neopixel library within the Arduino IDE. Using this standard allows for an easy programming of the LED lights without worrying about compatibility and the specific details about the components within the LED lights.

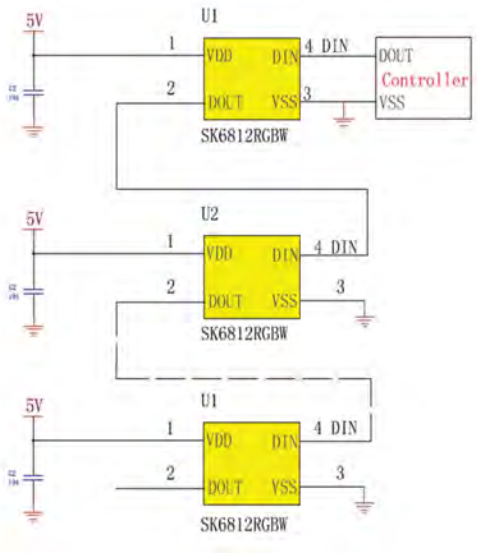


Figure 21: Connection Scheme [10]

Figure 22: Cabel layout in a real world setup

[8] (Intelligent LED Solutions, 2023)
[9] (Intelligent LED Solutions, 2023)
[10] (Intelligent LED Solutions, 2023)

5 The first user feedback

A first prototype was completed in time for the "Bright environments: Daylight in Sustainable Building Design Conference". User feedback was gathered through this opportunity and the model was modified accordingly.

This opportunity resulted in me assembling many analemmas for the first time. After testing all analemmas individually, all of them worked. However assembling all seven and plugging them into the Arduino introduced new problems. The analemma did not behave as expected and showed very random behaviour. After some investigation, the problem was identified to be the Arduino itself. The high number of analemmas forced the Arduino to process more data and control more LED lights. Even though the build was small enough at compilation time, it would produce as much control data for the LEDs at runtime, that the Arduino simply ran out of SRAM and started to behave unpredictably. Thus, a state with reduced functionality and no turning knob was created for the presentation at the conference. This problem was later addressed by choosing another more powerful version of the Arduino (Arduino Due instead of Arduino Uno Rev3). As a result of that problem, the knob control was not fully functional at the time of presentation and could thus not be tested with the audience.

Generally, the feedback of the people was very positive. This day showed, that visitors are generally attracted and interested by things which glow, and that the representation of the sun path as analemmas was not questioned by the daylight experts and thus must have been understandable. Very importantly, this day showed the need to provide a color legend and reminded me, that certain words specific to our field might not be in everybody's vocabulary and either need to be explained or omitted completely. As a general feedback though, the concept was understood well and no critical issues were pointed out.

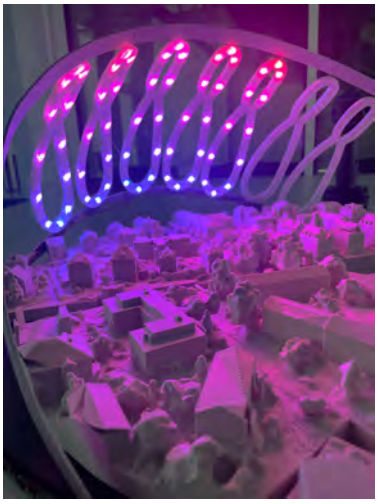


Figure 23: Temperature visualised for Zurich

Figure 24: Grid emission visualised for Zurich

Figure 25: The presentation to the audience

Figure 26: Assembly of the first bigger prototype

6 Details of the final design

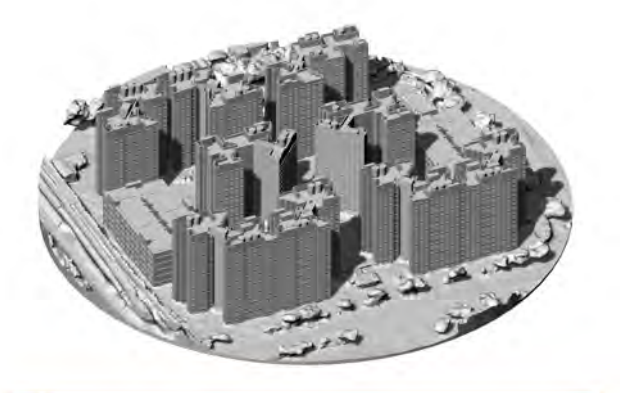
After the general concept was fixed (cf. *Section* 4.2) and the first presentation with user feedback was over (cf. *Section* 5), the production of the final model started. The 3D city model has been printed, a MDF base to host the model and to attach the sun path has been developped, the shape of the analemma has been reworked and the user interface/experience has been developed and improved to its final state, using the new Arduino Due, turning knobs and a LCD Display. As a last step a box has been developped to hide away the Arduino with the cables and to house the user interface and LCD screen (cf. *Section* 6.4).

This final version of the setup (cf. *Figure* 27, 45, and 46) is described to all details in the following sections.

Figure 27: Final Setup for Singapore

6.1 The 3D printed city model

To position the visualized sunpaths in a concrete geographical location, a model of the city at the studied places was printed out. These two models were based on a photogrammetry scan of google earth terrain done by Gregory Bianchi and then completed with hand modeled buildings by Chen Chiu (Rachel) and Christoph Waibel. For the prints a white PLA was used. To benefit from the high amount of detail a FDM print can offer, the layer height was set to 0.2mm with a nozzle diameter of 0.4mm. (For more details see the g-codes). The model was split into pieces as the dimensions of the printer are limited to 25cmx20cmx20cm and later on glued together again. The scale of the model was chosen, such that it is approximately 1:500 for both locations.



Singapore

Zurich

Figure 28: Data for the 3D printed model

Singapore

Zurich

Figure 29: Pictures of the finished printed model

6.2 The base

In addition to the model, a base was developed. The bases main purpose is to frame the white model with its black color and to provide hidden space and slits for cable management as well as to provide a solid volume to anker the metal support profiles of the sun path (cf. *Figure* 31).

A first experimentation with cutting MDF into a round shape by hand failed. To achieve the precision and the rather small thicknesses around the cable management slits (cf. *Figure* 32 and 33) and the border a laser cutter was used (executed by Schnittstelle Altstetten). Scaling the slits and the border to larger dimensions was not possible, as the size of the model was limited by the layout of the other content on the table. The base was glued together using wood glue. Later on additional holes next to the cable management slits were drilled to allow easier insertion of the cables into the slit. The models were fixed inside the base with velcro tape, such that they can still be removed, if changes need to be made to the cable or if a cable needs to be removed.

Singapore Zurich

Figure 30: 3D plan of the assambled base

Singapore Zurich

Figure 31: The finalised glued MDF base

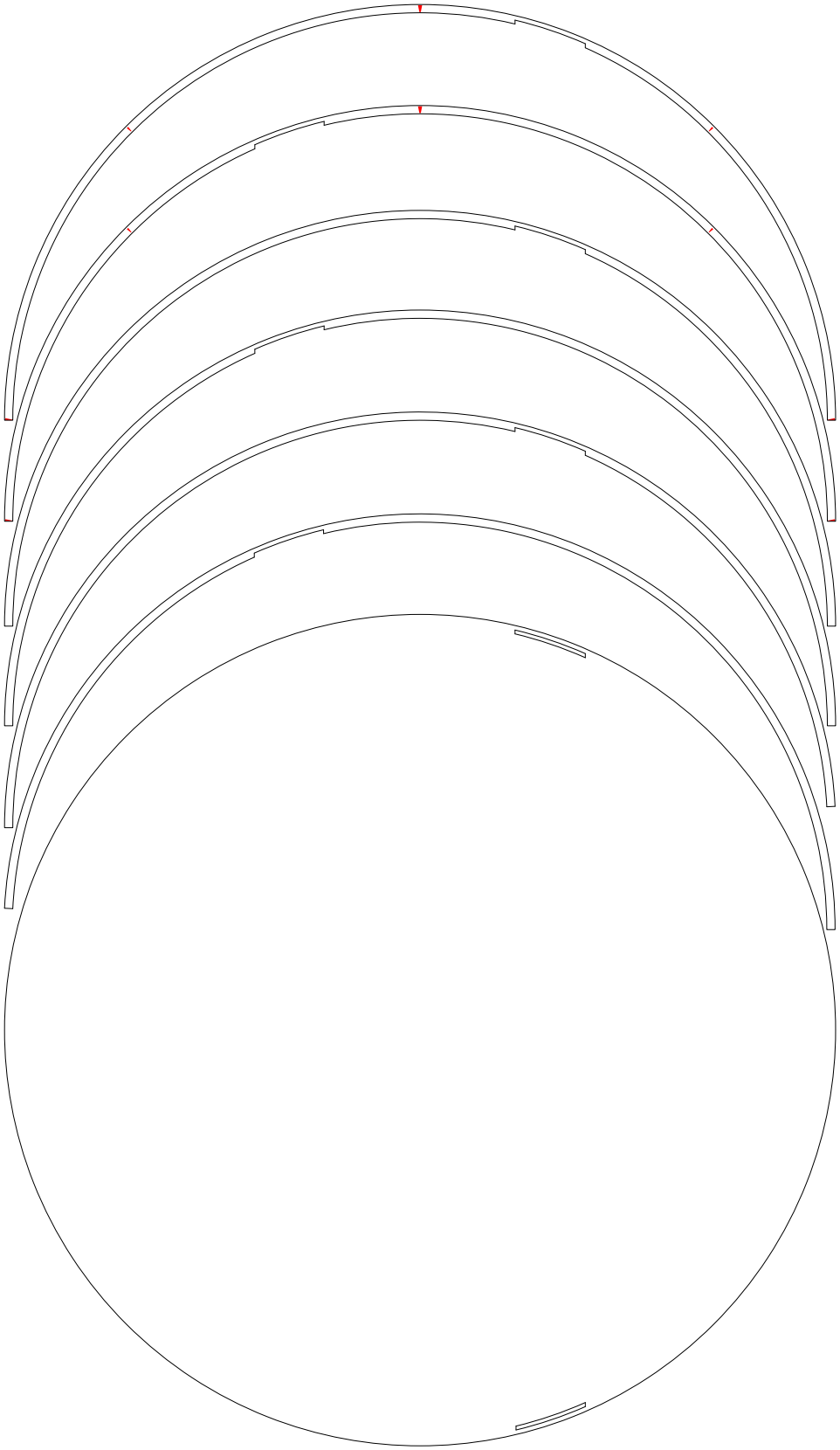


Figure 32: Cut scheme for the MDF laser cutting Singapore

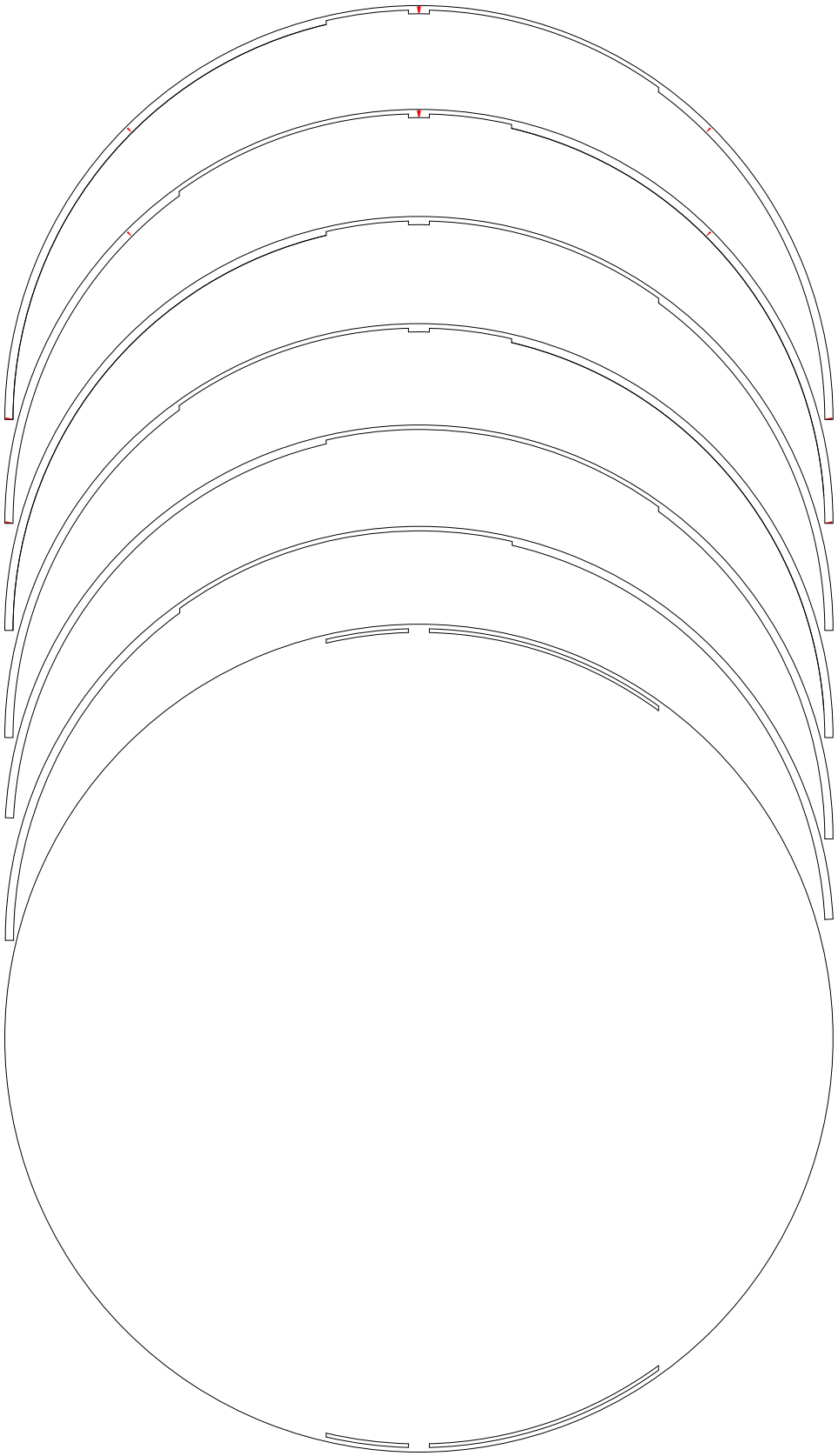


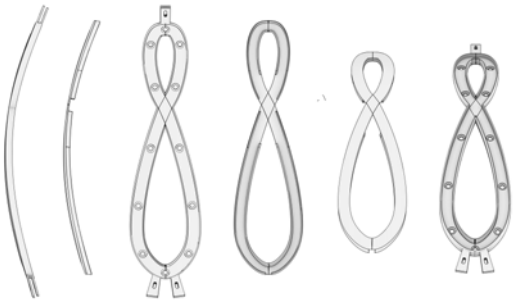
Figure 33: Cut scheme for the MDF laser cutting Zurich

6.3 The sunpath and the analemmas

The final representation of the sun path consists of two bent aluminum profiles marking the paths of the two solstices. They give stability to the whole system and act as mounting rails for the Analemmas which contain the LED lights.

As shown in *Section* 4.2.1 there has been some prototyping with differently shaped analemma housings. The final housing was produced in a transparent PLA filament. The final design consists of a housing with mounting latches (cf. *Figure* 35), which were fixed to the metal profiles attached to the base using glue (screwing them would also have been possible, as the latches have a hole prepared for a screw). The LED lights were soldered together with cables corresponding in length with the distances of the holes in the housing. After soldering, the LED lights were mounted inside the base using a double sided transparent scotch tape placed in the indent marking the LEDs position (cf. *Figure* 35). The LEDs were then taped down using a regular scotch tape. After the LED lights were mounted inside the base, this base was closed using a lid (*Figure* 35). The lid was designed to fit tightly into the base such that no adhesive is needed to hold it closed. That way it can be opened at any point in time, if repairs to the LED chain or a disassembly should be necessary. Designing the height of the analemmas very minimal, ensures the leds stay in place, since they are held down by the pressure of the lid.

For simplicity reason all analemmas have been approximated with the same shape, even though technically this does not correspond with reality. For the Zurich location the six analemmas on the side (first three in the morning and second 3 in the evening needed to be cut, as part of them are located below the horizon (cf. *Figure* 36)



Detail plan of the Analemma housing

Cut housings for Zurich

Figure 34: CAD plans of the Analemma housings

6.3.1 Cable management

A hole in the base as well as in the lid was was introduced in the final design to run the cables of the LEDs out of the housing. The cables were run along the underneath side of the metal profile, well hidden from views and subsequently through the slits in the base and out of the opening in the front of the base (cf. *Figure* 30 and 27). Then the cables were routed into the user control box (cf. *Section* 6.4) and connected to the respective sources using a system of "WAGO" connectors. (cf. *Section* 6.5).

Figure 35: Top view of the mounted analemmas)

Figure 36: Bottom view of the mounted analemmas

6.4 The user experience

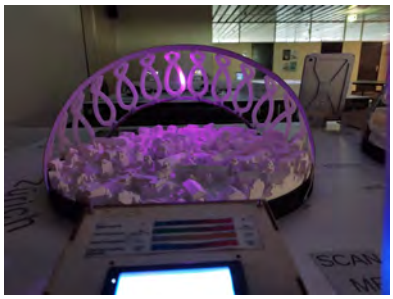
To allow the user to set the desired date and the data set he wants to see, an interface is needed. This will be created using physical turning knobs, as analog input is easier to handle with an Arduino micro controller than dealing with digital input on the serial port. Furthermore, an analog interface is easy to understand and use for almost everyone, as one just has to turn a knob.

For the aforementioned turning knobs three simple "Drehgeber" were used, which have 24 individual positions each. As a feedback (to see in which position the knob currently is set to) and to give more detailed information about the currently selected data a 4x20line LCD screen is used. To allow for a clear interface and a good cable management all these elements were cased using a plywood box (cf. *Figure* 38). The plywood box was lasercut and joind using glue and the finger joints. The lid of the box is removable and held in place by a set of screws and nuts in the corners.

This user interface allows the user to visualize the whole year, the sun path from morning to evening in a specific month or all the positions of the sun in the different months for one specific hour. Furthermore, if month and hours are set, the user can look at one specific LED. No matter what setting the knobs are in, the user is prompted with a minimal and a maximal data value for the currently active LEDs/sun positions. As such, if a user wants they can read out all data value, but can also have a good overview of the data. Basically leaving them optimal control over what is displayed. (cf. *Figure* 37) In addition to the LCD screen a printed legend was mounted, explaining the color code of the different modes, leading to a better distinguish-ability of the different modes and a better understanding of the user about what is shown. Using the same legend for Zurich as well as for Singapore allows cross comparability and might motivate some people to interact with both models trying to compare the different locations.

one month, all hours

all months, one hour



one month, one hour

Figure 37: Different possible knob settings

Figure 38: The horizontal coordinate system

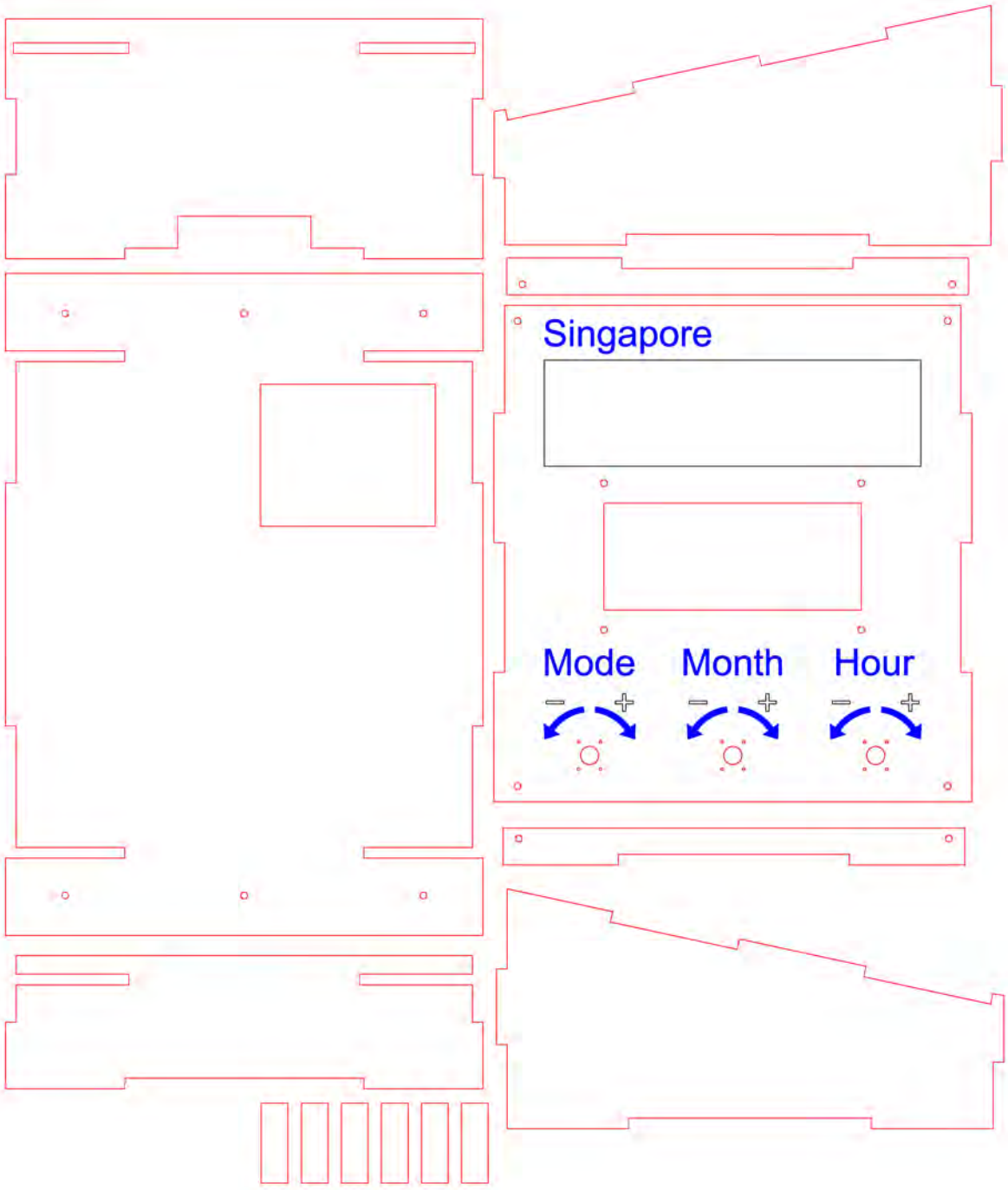


Figure 39: Cut scheme box Singapore

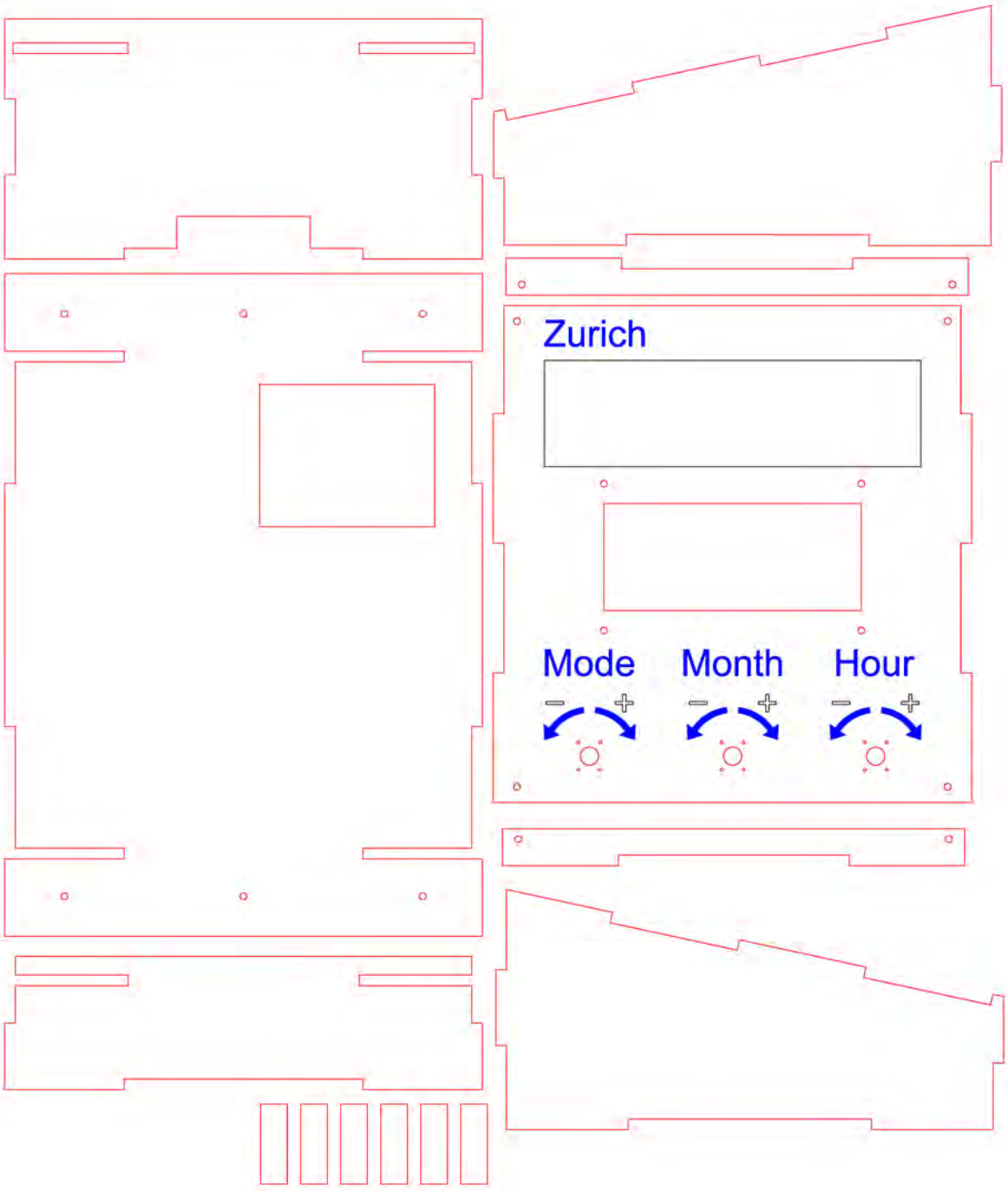


Figure 40: Cut scheme box Zurich

6.5 The Electronics and the Code

The heart of the installation is an Arduino Due. In some first attempts a standard Arduino was used. However, this lead to some errors and turned out to be infeasible due to the lack of enough RAM to drive the LEDs (cf. *Section* 5) and the availability of only two interuptable ports. As each turn of a knob needs to interrupt the code in its execution and register the new state of the knob, each of the three knobs need an interruptable port. Meaning three interuptable ports are needed. The Arduino due solves all the aforementioned problems. Being based on an ARM architecture, it has significantly more RAM and allows to have an interrupt on any of its ports. The LEDS were accessed using the libraries described in *Section* 4.2. The LCD mentioned in user experience was programmed using an I2C interface and the "LiquidCrystal I2C" library (which technically is not compatible with the Arduino Due, but worked very well nevertheless). The "Drehgeber" which were used as the turning knobs did not need any library, as they are just an electric circuit connecting three contacts in different combinations depending on the turn direction. These were implemented with a custom made code. The details about a possible cabling scheme of the Arduino can be seen below. The respective program code can be found in the *Annex A* and *Annex B*.

To ensure that the dome is always glowing and does not freeze due to a crash, a watchdog and a timer were introduced. The timer is reset every time a knob is turned. Should no knob be turned for about 3min and the timer reaches 9000, the installation will reset to a state were all LEDs are glowing. To randomly get one of the four modes after reset (sun position mode excepted), a random number is generated using a measurement of the signal at one of the analogue ports as a seed for the random function. This was done due to the lack of any better method to generate pseudo random numbers with the Arduino.

The code is structured the following. There is the setup routine, which initializes all the components, variables and flags. There is the loop function, which is called over and over again. In the loop, the variables "mode", "month" and "hour" are evaluated. The mode variable determines which routine is executed. The month variable determines the bounds forwarded to the respective routine (depending on the mode set) The chosen routine uses the "hour" and the bounds to set the LED lights to the respective information and requests the print routines to print the respective values on the display. As a last task, the loop checks, if the knobs were turned in the meantime (meaning the routine was interrupted). It reads the new state of the variables "mode", "month" and "hour" and then starts over.

If a knob is turned, when the loop is currently busy with another action than reading the knob state, the code will be interrupted, the variables "mode", "month" and "hour" will be set accordingly and the code continues where it was interrupted.

The hourly data was aggregated to "monthly" averages for every individual hour of the day and then formatted as arrays per hour in the from of
"$_{mode\_hour[Jan,\,Feb,\,Mar,\,Apr,\,May,\,Jun,\,Jul,\,Aug,\,Sep,\,Oct,\,Nov,\,Dec]}$"
As this is the most common order of the months from simulation outputs. However, the first LED in the LED chain is the led representing the winter solestice, meaning 21^{st} of December. Thus, in the code creating the control Data for the LEDs, the whole array was shifted by one position such that it corresponds with the physical position of the LEDs.

The color schemes for the different modes were created by creating the RGBW values with an affine function using the data values and the minimal and maximal data values of each data set. The affine function parameters were tuned in a way, such that the color scheme has an as good separation of the different colors as possible without loosing the cross comperability between the Zurich and the Singapore model.

For a more detailed insight consult the flow diagram below (cf. *Figure* 41 and the respective code sections in the *Annex A and Annex B*.

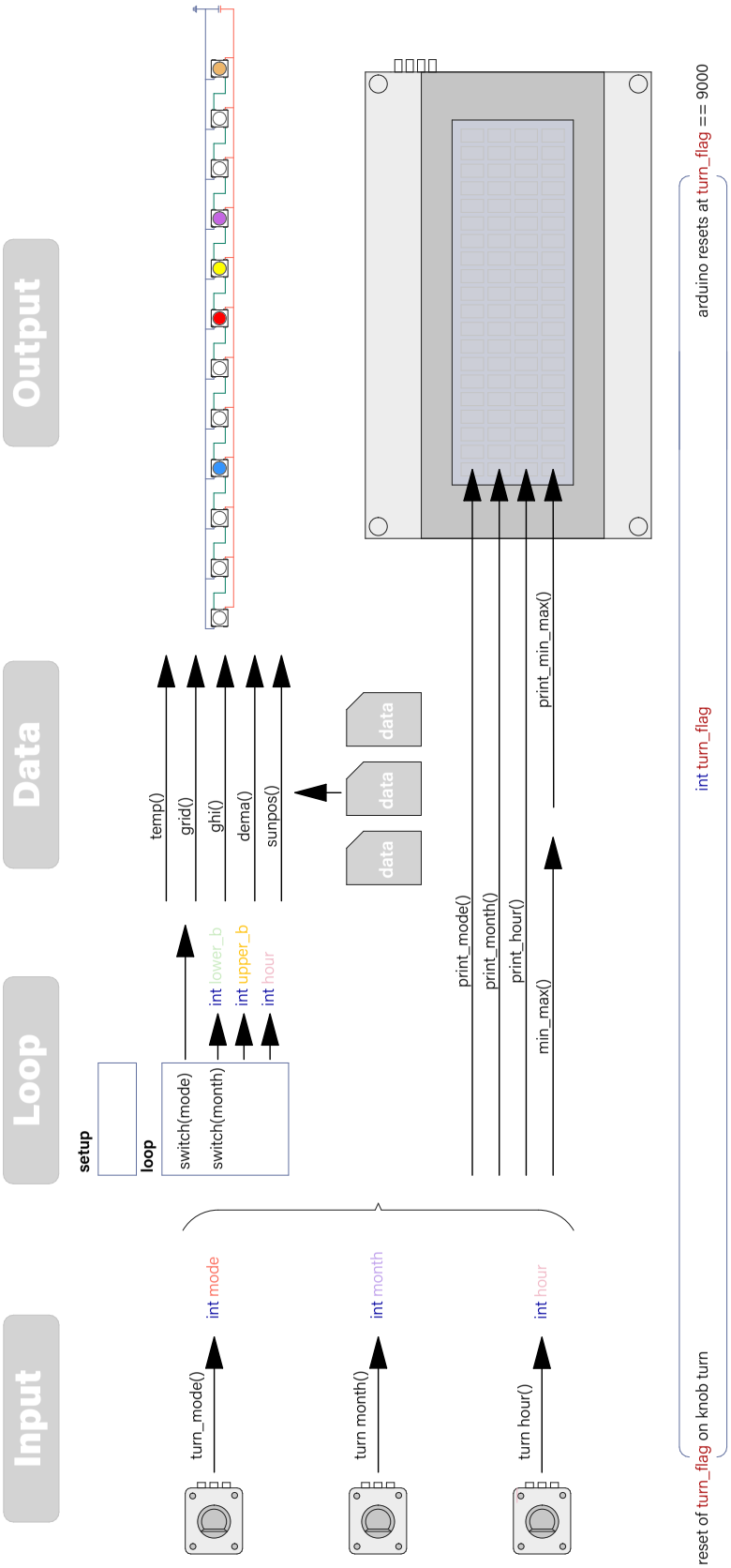


Figure 41: Connection Scheme Zurich

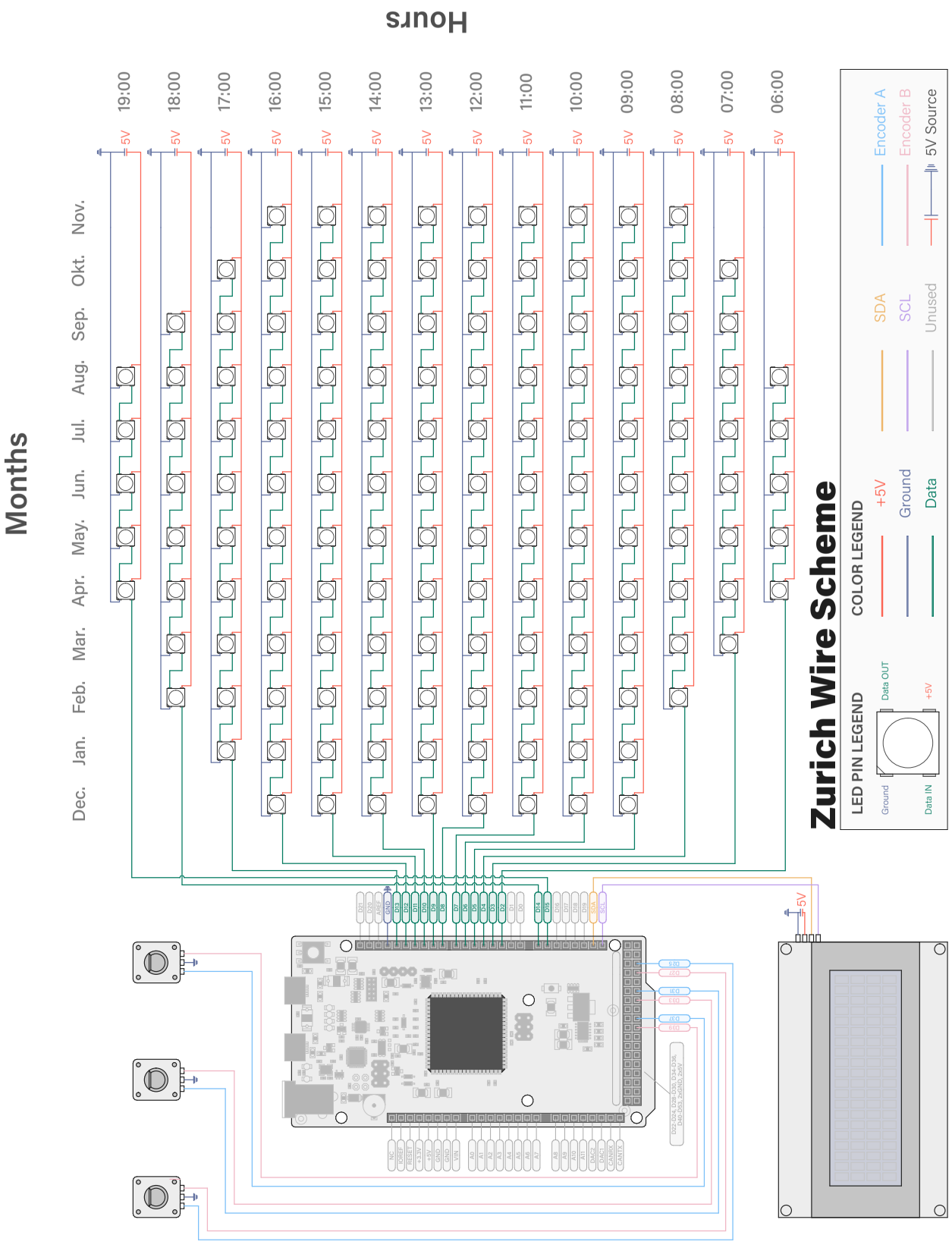


Figure 42: Connection Scheme Zurich

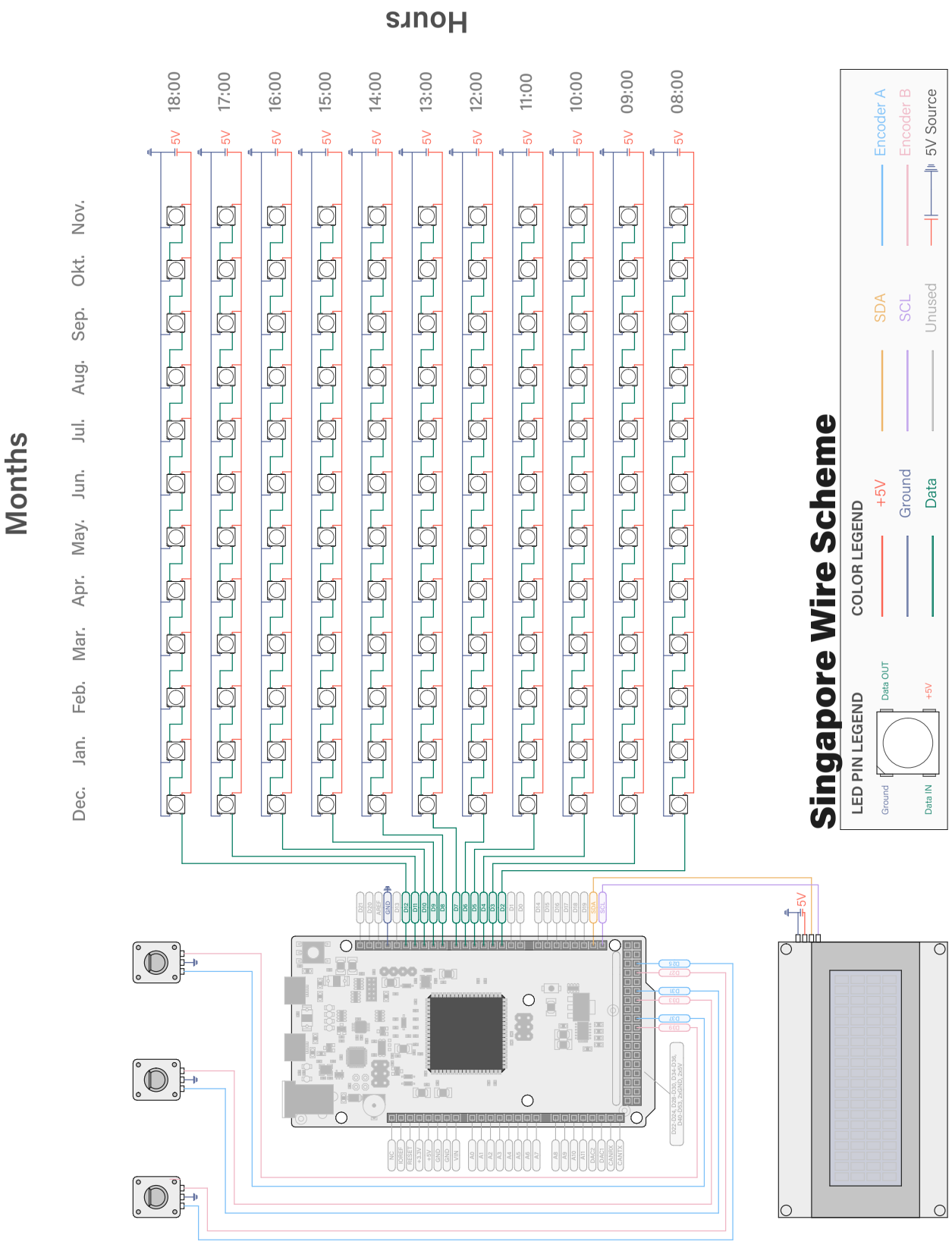


Figure 43: Connection Scheme Singapore

7 The final result

The final model was displayed during the "Future City Labaratory Conference" in the main building of ETH Zurich (cf. *Figure* 47). And can now be seen until May 2024 in the H-floor of the HIL on the Hönggerberg Campus. As far as I can tell, the model was received very good during the exhibition period and in my opinion the images below say it all.

Figure 44: The domes in different combinations of modes

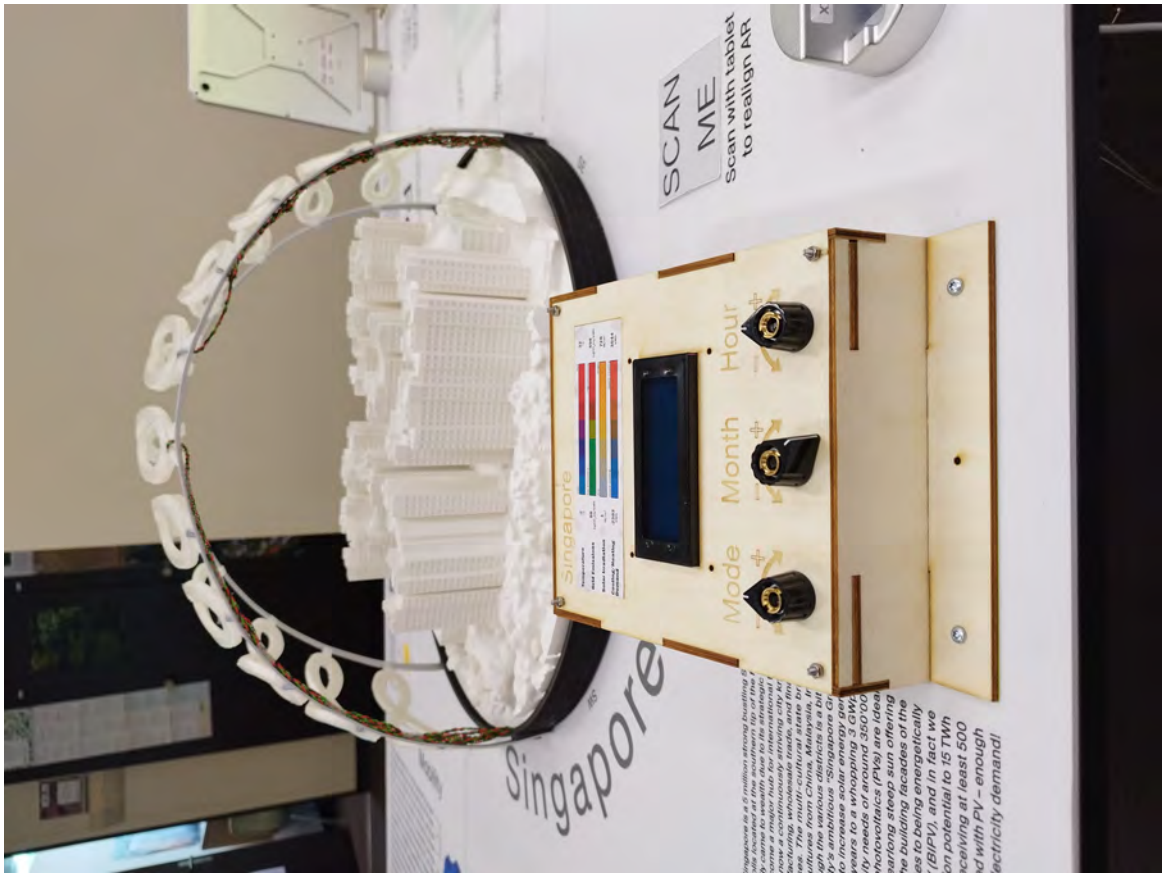


Figure 46: Final model Singapore

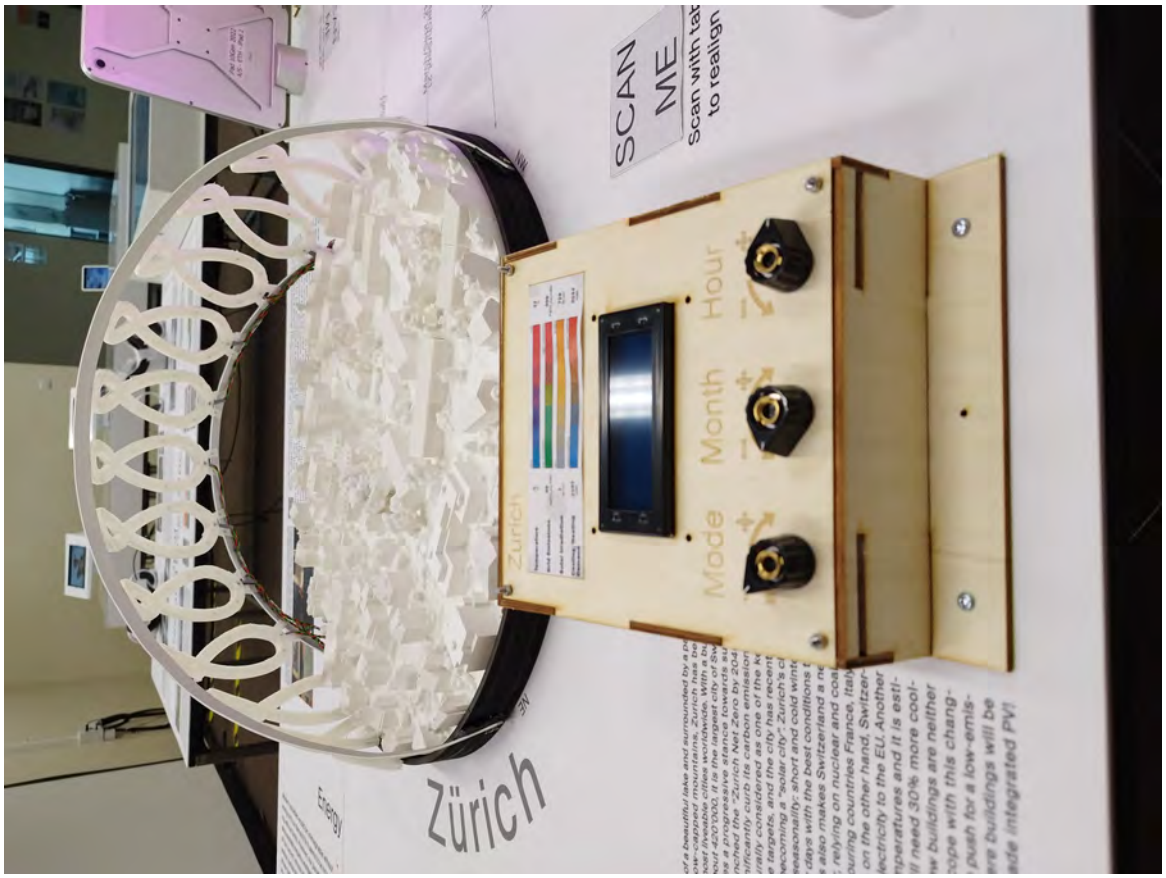


Figure 45: Final model Zurich

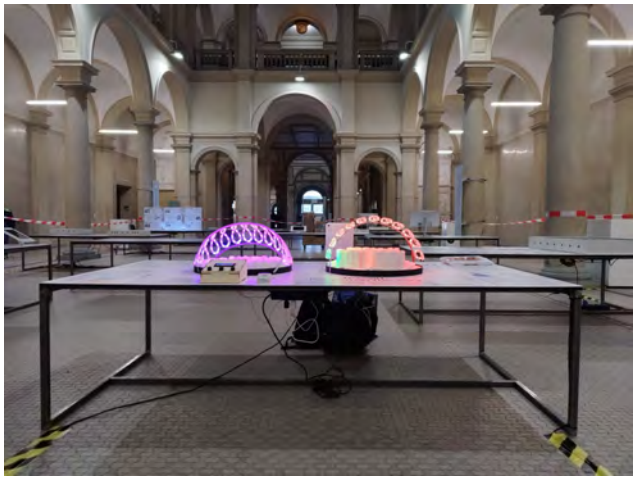


Figure 47: The exhibition setup in ETH main building

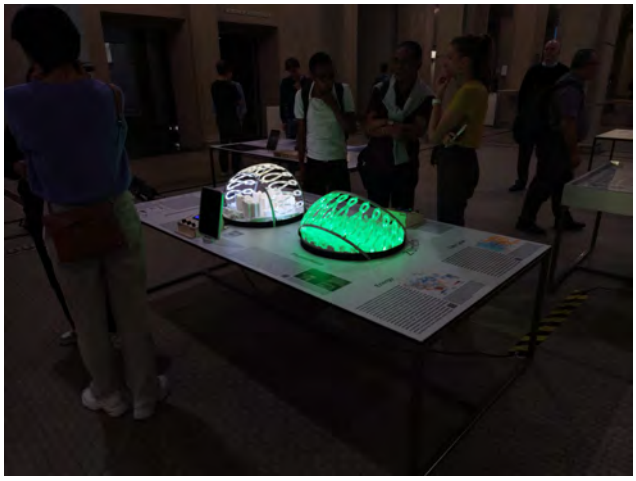


Figure 48: Exhibition in progress

Figure 49: Exhibition in progress

Figure 50: Model Zurich in white spotlight

Figure 51: Grid Emission Mode

Figure 52: Sun position mode

Figure 53: Heating/Cooling demand mode

Figure 54: Solar irradiation mode

Figure 55: Sun position mode

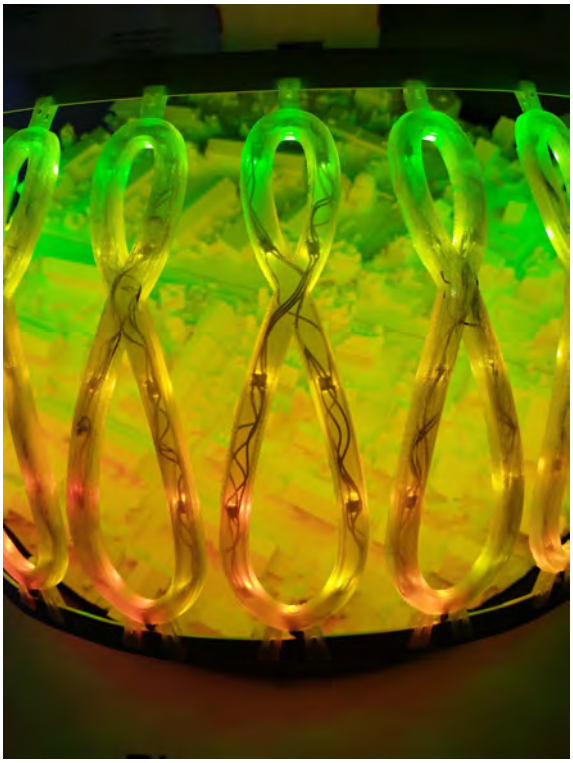


Figure 56: Grid emissions mode

References

[1] Daniel V. Schroeder. *Understanding Astronomy: The Sun and the Seasons*. Jan. 1, 2011. URL: https://physics.weber.edu/schroeder/ua/sunandseasons.html.

[2] Stanford Solar Center. *Viewing and Understanding the Analemma*. Jan. 1, 2022. URL: http://solar-center.stanford.edu/art/analemma.html.

[3] Laboratory of Integrated Performance in Design. *Facilities*. Jan. 1, 2023. URL: https://www.epfl.ch/labs/lipid/.

[4] Kyle Konis. "THE SOLAR HELIODON: PHYSICAL SIMULATION OF DYNAMIC DAYLIGHTING CONDITIONS IN SCALE ARCHITECTURAL MODELS FOR SUBJECTIVE AND OBJECTIVE HUMAN-FACTORS VALUATION". In: *Building Performance Analysis Conference and SimBuild co-organized by ASHRAE and IBPSA-U* (2020), pp. 111–118.

[5] Andreas Muxel. *Sensing the City*. Jan. 1, 2014. URL: https://www.andreasmuxel.com/teaching/sensing-the-city/.

[6] Arduino Team. *A community-made, Arduino-powered interactive town map*. Sept. 15, 2016. URL: https://blog.arduino.cc/2016/09/15/a-community-made-arduino-powered-interactive-town-map/.

[7] Szücs-Csillik, Iharka and Poputa, Dora. "The Astro-Biblio-Students program". In: 33 (Dec. 2015), pp. 101–111.

[8] Intelligent LED Solutions. *ILPL-K501-RGBW-SK105-01*. Jan. 1, 2023. URL: https://media.distrelec.com/Web/Downloads/_t/ds/ILPL-K501-RGBW-SK105-01_eng_tds.pdf.

Annex

Annex A: Code for the Singapore sun path visualisation

```
#include <Adafruit_NeoPixel.h>
#include <PGMWrap.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,20,4);


#define PIX_8 12
#define PIX_9 11
#define PIX_10 10
#define PIX_11 9
#define PIX_12 8
#define PIX_13 7
#define PIX_14 6
#define PIX_15 5
#define PIX_16 4
#define PIX_17 3
#define PIX_18 2


#define CONT_1 20
#define CONT_2 21
#define CONT_3 22

#define ENC_mode_A 24
#define ENC_mode_B 25
#define ENC_month_A 28
#define ENC_month_B 29
#define ENC_hour_A 32
#define ENC_hour_B 33

#define NUMPIXELS 12
#define DELAYVAL 100

//initialize pixles for the analemmas

Adafruit_NeoPixel pixels_8(NUMPIXELS, PIX_8, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_9(NUMPIXELS, PIX_9, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_10(NUMPIXELS, PIX_10, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_11(NUMPIXELS, PIX_11, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_12(NUMPIXELS, PIX_12, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_13(NUMPIXELS, PIX_13, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_14(NUMPIXELS, PIX_14, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_15(NUMPIXELS, PIX_15, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_16(NUMPIXELS, PIX_16, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_17(NUMPIXELS, PIX_17, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_18(NUMPIXELS, PIX_18, NEO_GRBW + NEO_KHZ800);
//initialize pixles for the control panel
Adafruit_NeoPixel control_1(NUMPIXELS, CONT_1, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel control_2(NUMPIXELS, CONT_2, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel control_3(NUMPIXELS, CONT_3, NEO_GRBW + NEO_KHZ800);



String unit_temp="C";
String unit_grid="kgCO2/kWh";
String unit_ghi="W/m2";
String unit_dema="kWh";
```

```
59 float temp_array[]={0,0,0,0,0,0,0,0,0,0,0};
60 volatile int mode=0;
61 volatile int month=12;
62 volatile int hour=11;
63 float minValue;
64 float maxValue;
65 int lower_b;
66 int upper_b;
67 int period=200;
68 int time_now;
69 int encoderPINA_mode = ENC_mode_A;
70 int encoderPINB_mode = ENC_mode_B;
71 int encoderPos_mode = 0;
72 int encoderPINALast_mode = LOW;
73 int encoderPINANow_mode = LOW;
74 int encoderPINA_month = ENC_month_A;
75 int encoderPINB_month = ENC_month_B;
76 int encoderPos_month = 0;
77 int encoderPINALast_month = LOW;
78 int encoderPINANow_month = LOW;
79 int encoderPINA_hour = ENC_hour_A;
80 int encoderPINB_hour = ENC_hour_B;
81 int encoderPos_hour = 0;
82 int encoderPINALast_hour = LOW;
83 int encoderPINANow_hour = LOW;
84 int a;
85 int aa;
86
87 // SINGAPORE GRID EMISSIONS
88 float avggrid_8[]={488,488,488,489,487,488,487,487,488,487,488,489};
89 float avggrid_9[]={492,491,492,492,492,492,492,491,493,492,492,492};
90 float avggrid_10[]={495,494,495,495,495,494,495,494,497,495,494,495};
91 float avggrid_11[]={496,495,496,496,495,495,497,495,497,496,494,496};
92 float avggrid_12[]={496,495,497,496,496,495,497,496,498,496,496,496};
93 float avggrid_13[]={496,495,497,496,496,495,497,496,498,496,496,496};
94 float avggrid_14[]={497,495,497,496,496,495,497,496,497,497,495,496};
95 float avggrid_15[]={497,496,497,497,496,496,498,496,497,497,496,496};
96 float avggrid_16[]={497,496,498,497,496,496,498,496,497,497,496,497};
97 float avggrid_17[]={497,496,498,497,496,496,498,496,497,497,496,497};
98 float avggrid_18[]={498,496,498,497,496,496,498,496,497,497,496,497};
99 float b_grid=0;
100 float max_grid=498;
101 float min_grid=300;
102 float max_grid_si=498;
103 float min_grid_si=487;
104
105 // SINGAPORE DEMAND
106
107
108 float avgdema_8[]={-279,-324,-716,-854,-1073,-925,-768,-842,-1003,-1052,-670,-498};
109 float avgdema_9[]={-770,-850,-942,-994,-1112,-999,-928,-966,-1055,-1025,-998,-862};
110 float avgdema_10[]={-875,-857,-817,-935,-1002,-937,-861,-827,-941,-931,-827,-802};
111 float avgdema_11[]={-1086,-1085,-1192,-1342,-1445,-1317,-1249,-1243,-1369,-1316,-1227,-1131};
112 float avgdema_12[]={-1432,-1448,-1535,-1735,-1824,-1670,-1596,-1610,-1746,-1685,-1630,-1459};
113 float avgdema_13[]={-1436,-1496,-1567,-1710,-1809,-1663,-1578,-1600,-1700,-1680,-1562,-1470};
114 float avgdema_14[]={-1058,-1164,-1223,-1292,-1386,-1237,-1173,-1180,-1286,-1271,-1160,-1052};
115 float avgdema_15[]={-829,-936,-1019,-1039,-1127,-995,-921,-942,-1048,-1011,-897,-798};
116 float avgdema_16[]={-755,-898,-939,-959,-1020,-900,-830,-860,-918,-922,-792,-723};
117 float avgdema_17[]={-958,-1023,-1053,-1129,-1198,-1074,-989,-1036,-1065,-1064,-926,-867};
118 float avgdema_18[]={-1839,-1871,-1911,-2055,-2158,-2050,-1935,-1972,-2004,-2006,-1827,-1748};
119 float g_dema=20;
120 float max_dema=-279;
121 float min_dema=-2383;
122 float max_dema_si=-279;
123 float min_dema_si=-2383;
124
125
126 // SINGAPORE GHI
127
128 float avgghi_8[]={161,162,174,179,189,170,173,177,213,229,182,162};
129 float avgghi_9[]={313,347,356,327,354,312,318,310,383,406,312,328};
```

```
130 float avgghi_10[]={495,518,529,467,492,476,450,437,508,537,421,456};
131 float avgghi_11[]={588,594,616,552,514,555,534,513,579,583,494,527};
132 float avgghi_12[]={646,717,700,639,635,589,582,576,638,609,537,520};
133 float avgghi_13[]={645,728,698,696,603,598,564,592,629,572,576,569};
134 float avgghi_14[]={604,693,640,626,538,539,540,587,565,512,494,494};
135 float avgghi_15[]={509,601,559,494,457,458,498,536,445,421,405,384};
136 float avgghi_16[]={390,429,413,361,327,350,392,409,322,282,305,294};
137 float avgghi_17[]={224,267,243,218,184,195,235,233,181,150,144,168};
138 float avgghi_18[]={80,103,81,69,50,62,82,73,32,12,10,27};
139 float g_ghi=200;
140 float max_ghi=728;
141 float min_ghi=1;
142 float max_ghi_si=728;
143 float min_ghi_si=1;
144
145
146 // SINGAPORE TEMP
147
148 float avgtemp_8[]={25,25,25,25,26,26,26,26,25,26,25,25};
149 float avgtemp_9[]={25,25,26,26,27,27,27,27,26,27,26,26};
150 float avgtemp_10[]={26,27,27,27,28,28,28,28,27,28,26,27};
151 float avgtemp_11[]={27,28,28,28,29,29,29,29,28,29,27,28};
152 float avgtemp_12[]={28,28,29,29,30,30,29,29,29,30,28,28};
153 float avgtemp_13[]={29,29,30,30,31,30,30,30,30,30,29,29};
154 float avgtemp_14[]={29,30,30,30,31,31,30,30,30,31,29,29};
155 float avgtemp_15[]={29,30,31,31,32,31,31,31,31,31,30,30};
156 float avgtemp_16[]={29,30,31,31,32,31,31,31,31,31,30,29};
157 float avgtemp_17[]={29,30,31,31,32,31,31,31,30,31,30,29};
158 float avgtemp_18[]={29,30,30,30,31,30,30,30,30,30,29,29};
159 float g_temp=0;
160 float max_temp=32;
161 float min_temp=20;
162 float max_temp_si=32;
163 float min_temp_si=25;
164
165 int r_sun=100;
166 int g_sun=100;
167 int b_sun=100;
168 int w_sun=200;
169
170 int reset_timeout=9000;
171 int counter=0;
172 int turn_flag=0;
173
174 void setup() {
175   watchdogSetup();
176   watchdogEnable(10000);
177   Serial.begin (115200);
178   //initialize LCD
179   lcd.init();
180   Wire.setClock(400000);
181   lcd.clear();
182   lcd.backlight();
183   //initialize Pixels
184
185   pixels_8.begin();
186   pixels_9.begin();
187   pixels_10.begin();
188   pixels_11.begin();
189   pixels_12.begin();
190   pixels_13.begin();
191   pixels_14.begin();
192   pixels_15.begin();
193   pixels_16.begin();
194   pixels_17.begin();
195   pixels_18.begin();
196   pinMode (encoderPINA_mode, INPUT_PULLUP);
197   pinMode (encoderPINB_mode, INPUT_PULLUP);
198   pinMode (encoderPINA_month, INPUT_PULLUP);
199   pinMode (encoderPINB_month, INPUT_PULLUP);
200   pinMode (encoderPINA_hour, INPUT_PULLUP);
```

```
201   pinMode (encoderPINB_hour, INPUT_PULLUP);
202   attachInterrupt(digitalPinToInterrupt(encoderPINA_month), turn_month, CHANGE);
203   attachInterrupt(digitalPinToInterrupt(encoderPINA_hour), turn_hour, CHANGE);
204   attachInterrupt(digitalPinToInterrupt(encoderPINA_mode), turn_mode, CHANGE);
205   randomSeed(analogRead(A2));
206   mode=random(0,4);
207   print_mode();
208   print_month();
209   print_hour();
210   // put your setup code here, to run once:
211
212 }
213
214
215 void loop() {
216   watchdogReset();
217   if (turn_flag==1){
218     counter=0;
219     turn_flag=0;
220   }
221   if( counter  >= reset_timeout && turn_flag==0 ){
222     rstc_start_software_reset(RSTC);
223   }
224   counter++;
225
226   switch (month) {
227   case 0:
228   case 1:
229   case 2:
230   case 3:
231   case 4:
232   case 5:
233   case 6:
234   case 7:
235   case 8:
236   case 9:
237   case 10:
238   case 11:
239   upper_b=month+1;
240   lower_b=month;
241   break;
242   case 12:
243   upper_b=12;
244   lower_b=0;
245   break;
246   }
247
248
249   //mode switch
250   switch (mode) {
251   case 0:
252   //clear_pix();
253   temp();
254   show();
255   break;
256   case 1:
257   //clear_pix();
258   grid();
259   show();
260   break;
261   case 2:
262   //clear_pix();
263   ghi();
264   show();
265   break;
266   case 3:
267   //clear_pix();
268   dema();
269   show();
270   break;
271   case 4:
```

```
272   sunpos();
273   show();
274   break;
275   }
276
277
278
279   turn_mode();
280   turn_month();
281   turn_hour();
282   //clear_pix();
283
284
285 }
286   void ghi(){
287
288     for (int i = lower_b; i < upper_b; i++) {
289       if(i==0){
290         a=11;
291       }else{
292         a=i-1;
293       }
294       switch(hour){
295         case 0:
296           pixels_8.setPixelColor(i,120+125*(avgghi_8[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,170*(
      max_ghi-avgghi_8[a])/(max_ghi-min_ghi),0);
297           break;
298         case 1:
299           pixels_9.setPixelColor(i,120+125*(avgghi_9[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,170*(
      max_ghi-avgghi_9[a])/(max_ghi-min_ghi),0);
300           break;
301         case 2:
302           pixels_10.setPixelColor(i,120+125*(avgghi_10[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_10[a])/(max_ghi-min_ghi),0);
303           break;
304         case 3:
305           pixels_11.setPixelColor(i,120+125*(avgghi_11[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_11[a])/(max_ghi-min_ghi),0);
306           break;
307         case 4:
308           pixels_12.setPixelColor(i,120+125*(avgghi_12[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_12[a])/(max_ghi-min_ghi),0);
309           break;
310         case 5:
311           pixels_13.setPixelColor(i,120+125*(avgghi_13[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_13[a])/(max_ghi-min_ghi),0);
312           break;
313         case 6:
314           pixels_14.setPixelColor(i,120+125*(avgghi_14[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_14[a])/(max_ghi-min_ghi),0);
315           break;
316         case 7:
317           pixels_15.setPixelColor(i,120+125*(avgghi_15[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_15[a])/(max_ghi-min_ghi),0);
318           break;
319         case 8:
320           pixels_16.setPixelColor(i,120+125*(avgghi_16[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_16[a])/(max_ghi-min_ghi),0);
321           break;
322         case 9:
323           pixels_17.setPixelColor(i,120+125*(avgghi_17[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_17[a])/(max_ghi-min_ghi),0);
324           break;
325         case 10:
326           pixels_18.setPixelColor(i,120+125*(avgghi_18[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,170*(max_ghi-avgghi_18[a])/(max_ghi-min_ghi),0);
327           break;
328         case 11:
329
330           pixels_8.setPixelColor(i,120+125*(avgghi_8[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,170*(
      max_ghi-avgghi_8[a])/(max_ghi-min_ghi),0);
```

```
331             pixels_9.setPixelColor(i,120+125*(avgghi_9[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,170*(
    max_ghi-avgghi_9[a])/(max_ghi-min_ghi),0);
332             pixels_10.setPixelColor(i,120+125*(avgghi_10[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_10[a])/(max_ghi-min_ghi),0);
333             pixels_11.setPixelColor(i,120+125*(avgghi_11[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_11[a])/(max_ghi-min_ghi),0);
334             pixels_12.setPixelColor(i,120+125*(avgghi_12[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_12[a])/(max_ghi-min_ghi),0);
335             pixels_13.setPixelColor(i,120+125*(avgghi_13[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_13[a])/(max_ghi-min_ghi),0);
336             pixels_14.setPixelColor(i,120+125*(avgghi_14[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_14[a])/(max_ghi-min_ghi),0);
337             pixels_15.setPixelColor(i,120+125*(avgghi_15[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_15[a])/(max_ghi-min_ghi),0);
338             pixels_16.setPixelColor(i,120+125*(avgghi_16[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_16[a])/(max_ghi-min_ghi),0);
339             pixels_17.setPixelColor(i,120+125*(avgghi_17[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_17[a])/(max_ghi-min_ghi),0);
340             pixels_18.setPixelColor(i,120+125*(avgghi_18[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,170*(max_ghi-avgghi_18[a])/(max_ghi-min_ghi),0);
341
342             break;
343     }
344     }
345   }
346
347   void temp(){
348     for (int i = lower_b; i < upper_b; i++) {
349       if(i==0){
350         a=11;
351       }else{
352         a=i-1;
353       }
354       switch(hour){
355         case 0:
356           pixels_8.setPixelColor(i,255*(avgtemp_8[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_8[a])/(max_temp-min_temp),0);
357           break;
358         case 1:
359           pixels_9.setPixelColor(i,255*(avgtemp_9[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_9[a])/(max_temp-min_temp),0);
360           break;
361         case 2:
362           pixels_10.setPixelColor(i,255*(avgtemp_10[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_10[a])/(max_temp-min_temp),0);
363           break;
364         case 3:
365           pixels_11.setPixelColor(i,255*(avgtemp_11[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_11[a])/(max_temp-min_temp),0);
366           break;
367         case 4:
368           pixels_12.setPixelColor(i,255*(avgtemp_12[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_12[a])/(max_temp-min_temp),0);
369           break;
370         case 5:
371           pixels_13.setPixelColor(i,255*(avgtemp_13[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_13[a])/(max_temp-min_temp),0);
372           break;
373         case 6:
374           pixels_14.setPixelColor(i,255*(avgtemp_14[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_14[a])/(max_temp-min_temp),0);
375           break;
376         case 7:
377           pixels_15.setPixelColor(i,255*(avgtemp_15[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_15[a])/(max_temp-min_temp),0);
378           break;
379         case 8:
380           pixels_16.setPixelColor(i,255*(avgtemp_16[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_16[a])/(max_temp-min_temp),0);
381           break;
382         case 9:
```

```
383           pixels_17.setPixelColor(i,255*(avgtemp_17[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_17[a])/(max_temp-min_temp),0);
384           break;
385         case 10:
386           pixels_18.setPixelColor(i,255*(avgtemp_18[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_18[a])/(max_temp-min_temp),0);
387           break;
388         case 11:
389
390           pixels_8.setPixelColor(i,255*(avgtemp_8[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
      max_temp-avgtemp_8[a])/(max_temp-min_temp),0);
391           pixels_9.setPixelColor(i,255*(avgtemp_9[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
      max_temp-avgtemp_9[a])/(max_temp-min_temp),0);
392           pixels_10.setPixelColor(i,255*(avgtemp_10[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_10[a])/(max_temp-min_temp),0);
393           pixels_11.setPixelColor(i,255*(avgtemp_11[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_11[a])/(max_temp-min_temp),0);
394           pixels_12.setPixelColor(i,255*(avgtemp_12[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_12[a])/(max_temp-min_temp),0);
395           pixels_13.setPixelColor(i,255*(avgtemp_13[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_13[a])/(max_temp-min_temp),0);
396           pixels_14.setPixelColor(i,255*(avgtemp_14[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_14[a])/(max_temp-min_temp),0);
397           pixels_15.setPixelColor(i,255*(avgtemp_15[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_15[a])/(max_temp-min_temp),0);
398           pixels_16.setPixelColor(i,255*(avgtemp_16[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_16[a])/(max_temp-min_temp),0);
399           pixels_17.setPixelColor(i,255*(avgtemp_17[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_17[a])/(max_temp-min_temp),0);
400           pixels_18.setPixelColor(i,255*(avgtemp_18[a]-min_temp)/(max_temp-min_temp), g_temp
      ,255*(max_temp-avgtemp_18[a])/(max_temp-min_temp),0);
401
402           break;
403       }
404     }
405   }
406
407   void dema(){
408     for (int i = lower_b; i < upper_b; i++) {
409       if(i==0){
410         a=11;
411       }else{
412         a=i-1;
413       }
414       switch(hour){
415
416         case 0:
417           pixels_8.setPixelColor(i,50*(avgdema_8[a]-min_dema)/(max_dema-min_dema), g_dema
      ,50+205*(max_dema-avgdema_8[a])/(max_dema-min_dema),0);
418           break;
419         case 1:
420           pixels_9.setPixelColor(i,50*(avgdema_9[a]-min_dema)/(max_dema-min_dema), g_dema
      ,50+205*(max_dema-avgdema_9[a])/(max_dema-min_dema),0);
421           break;
422         case 2:
423           pixels_10.setPixelColor(i,50*(avgdema_10[a]-min_dema)/(max_dema-min_dema), g_dema
      ,50+205*(max_dema-avgdema_10[a])/(max_dema-min_dema),0);
424           break;
425         case 3:
426           pixels_11.setPixelColor(i,50*(avgdema_11[a]-min_dema)/(max_dema-min_dema), g_dema
      ,50+205*(max_dema-avgdema_11[a])/(max_dema-min_dema),0);
427           break;
428         case 4:
429           pixels_12.setPixelColor(i,50*(avgdema_12[a]-min_dema)/(max_dema-min_dema), g_dema
      ,50+205*(max_dema-avgdema_12[a])/(max_dema-min_dema),0);
430           break;
431         case 5:
432           pixels_13.setPixelColor(i,50*(avgdema_13[a]-min_dema)/(max_dema-min_dema), g_dema
      ,50+205*(max_dema-avgdema_13[a])/(max_dema-min_dema),0);
433           break;
434         case 6:
```

```
435             pixels_14.setPixelColor(i,50*(avgdema_14[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_14[a])/(max_dema-min_dema),0);
436             break;
437           case 7:
438             pixels_15.setPixelColor(i,50*(avgdema_15[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_15[a])/(max_dema-min_dema),0);
439             break;
440           case 8:
441             pixels_16.setPixelColor(i,50*(avgdema_16[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_16[a])/(max_dema-min_dema),0);
442             break;
443           case 9:
444             pixels_17.setPixelColor(i,50*(avgdema_17[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_17[a])/(max_dema-min_dema),0);
445             break;
446           case 10:
447             pixels_18.setPixelColor(i,50*(avgdema_18[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_18[a])/(max_dema-min_dema),0);
448             break;
449           case 11:
450
451             pixels_8.setPixelColor(i,50*(avgdema_8[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_8[a])/(max_dema-min_dema),0);
452             pixels_9.setPixelColor(i,50*(avgdema_9[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_9[a])/(max_dema-min_dema),0);
453             pixels_10.setPixelColor(i,50*(avgdema_10[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_10[a])/(max_dema-min_dema),0);
454             pixels_11.setPixelColor(i,50*(avgdema_11[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_11[a])/(max_dema-min_dema),0);
455             pixels_12.setPixelColor(i,50*(avgdema_12[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_12[a])/(max_dema-min_dema),0);
456             pixels_13.setPixelColor(i,50*(avgdema_13[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_13[a])/(max_dema-min_dema),0);
457             pixels_14.setPixelColor(i,50*(avgdema_14[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_14[a])/(max_dema-min_dema),0);
458             pixels_15.setPixelColor(i,50*(avgdema_15[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_15[a])/(max_dema-min_dema),0);
459             pixels_16.setPixelColor(i,50*(avgdema_16[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_16[a])/(max_dema-min_dema),0);
460             pixels_17.setPixelColor(i,50*(avgdema_17[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_17[a])/(max_dema-min_dema),0);
461             pixels_18.setPixelColor(i,50*(avgdema_18[a]-min_dema)/(max_dema-min_dema), g_dema
    ,50+205*(max_dema-avgdema_18[a])/(max_dema-min_dema),0);
462             break;
463       }
464     }
465   }
466
467 void grid(){
468     for (int i = lower_b; i < upper_b; i++) {
469       if(i==0){
470         a=11;
471       }else{
472         a=i-1;
473       }
474         switch (hour) {
475         case 0:
476           pixels_8.setPixelColor(i,255*(avggrid_8[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_8[a])/(max_grid-min_grid),b_grid,0);
477           break;
478         case 1:
479           pixels_9.setPixelColor(i,255*(avggrid_9[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_9[a])/(max_grid-min_grid),b_grid,0);
480           break;
481         case 2:
482           pixels_10.setPixelColor(i,255*(avggrid_10[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_10[a])/(max_grid-min_grid),b_grid,0);
483           break;
484         case 3:
485           pixels_11.setPixelColor(i,255*(avggrid_11[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_11[a])/(max_grid-min_grid),b_grid,0);
```

```
486           break;
487         case 4:
488           pixels_12.setPixelColor(i,255*(avggrid_12[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_12[a])/(max_grid-min_grid),b_grid,0);
489           break;
490         case 5:
491           pixels_13.setPixelColor(i,255*(avggrid_13[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_13[a])/(max_grid-min_grid),b_grid,0);
492           break;
493         case 6:
494           pixels_14.setPixelColor(i,255*(avggrid_14[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_14[a])/(max_grid-min_grid),b_grid,0);
495           break;
496         case 7:
497           pixels_15.setPixelColor(i,255*(avggrid_15[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_15[a])/(max_grid-min_grid),b_grid,0);
498           break;
499         case 8:
500           pixels_16.setPixelColor(i,255*(avggrid_16[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_16[a])/(max_grid-min_grid),b_grid,0);
501           break;
502         case 9:
503           pixels_17.setPixelColor(i,255*(avggrid_17[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_17[a])/(max_grid-min_grid),b_grid,0);
504           break;
505         case 10:
506           pixels_18.setPixelColor(i,255*(avggrid_18[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_18[a])/(max_grid-min_grid),b_grid,0);
507           break;
508         case 11:
509 
510           pixels_8.setPixelColor(i,255*(avggrid_8[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_8[a])/(max_grid-min_grid),b_grid,0);
511           pixels_9.setPixelColor(i,255*(avggrid_9[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_9[a])/(max_grid-min_grid),b_grid,0);
512           pixels_10.setPixelColor(i,255*(avggrid_10[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_10[a])/(max_grid-min_grid),b_grid,0);
513           pixels_11.setPixelColor(i,255*(avggrid_11[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_11[a])/(max_grid-min_grid),b_grid,0);
514           pixels_12.setPixelColor(i,255*(avggrid_12[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_12[a])/(max_grid-min_grid),b_grid,0);
515           pixels_13.setPixelColor(i,255*(avggrid_13[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_13[a])/(max_grid-min_grid),b_grid,0);
516           pixels_14.setPixelColor(i,255*(avggrid_14[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_14[a])/(max_grid-min_grid),b_grid,0);
517           pixels_15.setPixelColor(i,255*(avggrid_15[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_15[a])/(max_grid-min_grid),b_grid,0);
518           pixels_16.setPixelColor(i,255*(avggrid_16[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_16[a])/(max_grid-min_grid),b_grid,0);
519           pixels_17.setPixelColor(i,255*(avggrid_17[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_17[a])/(max_grid-min_grid),b_grid,0);
520           pixels_18.setPixelColor(i,255*(avggrid_18[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_18[a])/(max_grid-min_grid),b_grid,0);
521 
522           break;
523 
524 
525         }
526 
527     }
528   }
529   void sunpos(){
530     for (int i = lower_b; i < upper_b; i++) {
531       switch (hour) {
532 
533         case 0:
534           pixels_8.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
535           break;
536         case 1:
537           pixels_9.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
538           break;
```

```
539           case 2:
540             pixels_10.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
541             break;
542           case 3:
543             pixels_11.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
544             break;
545           case 4:
546             pixels_12.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
547             break;
548           case 5:
549             pixels_13.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
550             break;
551           case 6:
552             pixels_14.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
553             break;
554           case 7:
555             pixels_15.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
556             break;
557           case 8:
558             pixels_16.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
559             break;
560           case 9:
561             pixels_17.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
562             break;
563           case 10:
564             pixels_18.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
565             break;
566           case 11:
567
568             pixels_8.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
569             pixels_9.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
570             pixels_10.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
571             pixels_11.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
572             pixels_12.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
573             pixels_13.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
574             pixels_14.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
575             pixels_15.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
576             pixels_16.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
577             pixels_17.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
578             pixels_18.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
579             break;
580       }
581
582
583
584
585
586     }
587   }
588   void wait(){
589     while(millis() < time_now + period){
590         //wait approx. [period] ms
591           }
592     time_now = millis();
593     show();
594   }
595
596   void show(){
597
598     pixels_8.show();
599     pixels_9.show();
600     pixels_10.show();
601     pixels_11.show();
602     pixels_12.show();
603     pixels_13.show();
604     pixels_14.show();
605     pixels_15.show();
606     pixels_16.show();
607     pixels_17.show();
608     pixels_18.show();
609   }
```

```
610
611 void clear_pix(){
612
613     pixels_8.clear();
614     pixels_9.clear();
615     pixels_10.clear();
616     pixels_11.clear();
617     pixels_12.clear();
618     pixels_13.clear();
619     pixels_14.clear();
620     pixels_15.clear();
621     pixels_16.clear();
622     pixels_17.clear();
623     pixels_18.clear();
624     show();
625 }
626
627 void turn_mode(){
628   encoderPINANow_mode = digitalRead(encoderPINA_mode);
629   if ((encoderPINALast_mode == HIGH) && (encoderPINANow_mode == LOW)) {
630     if (digitalRead(encoderPINB_mode) == HIGH) {
631       if (encoderPos_mode<4) {
632         encoderPos_mode++;
633       } else {
634       encoderPos_mode=0;
635       }
636     } else {
637       if(encoderPos_mode>0){
638         encoderPos_mode--;
639       } else if (encoderPos_mode==0) {
640         encoderPos_mode=4;
641       }
642     }
643
644     clear_pix();
645     clearLCDLine(0);
646     turn_flag=1;
647     mode=encoderPos_mode;
648     print_mode();
649   }
650   encoderPINALast_mode = encoderPINANow_mode;
651 }
652
653 void turn_month(){
654   encoderPINANow_month = digitalRead(encoderPINA_month);
655   if ((encoderPINALast_month == HIGH) && (encoderPINANow_month == LOW)) {
656     if (digitalRead(encoderPINB_month) == HIGH) {
657       if (encoderPos_month<12) {
658         encoderPos_month++;
659       } else {
660       encoderPos_month=0;
661       }
662     } else {
663       if(encoderPos_month>0){
664         encoderPos_month--;
665       } else if (encoderPos_month==0) {
666         encoderPos_month=12;
667       }
668     }
669
670     clear_pix();
671     clearLCDLine(1);
672     turn_flag=1;
673     month=encoderPos_month;
674     print_month();
675   }
676   encoderPINALast_month = encoderPINANow_month;
677 }
678
679 void turn_hour(){
680   encoderPINANow_hour = digitalRead(encoderPINA_hour);
```

```
681   if ((encoderPINALast_hour == HIGH) && (encoderPINANow_hour == LOW)) {
682     if (digitalRead(encoderPINB_hour) == HIGH) {
683       if (encoderPos_hour<11) {
684         encoderPos_hour++;
685       } else {
686       encoderPos_hour=0;
687       }
688     } else {
689       if(encoderPos_hour>0){
690         encoderPos_hour--;
691       } else if (encoderPos_hour==0) {
692         encoderPos_hour=11;
693       }
694     }
695     clear_pix();
696     clearLCDLine(2);
697     turn_flag=1;
698     Serial.println(encoderPos_hour);
699     hour=encoderPos_hour;
700     print_hour();
701   }
702   encoderPINALast_hour = encoderPINANow_hour;
703 }
704
705 void print_month(){
706   lcd.setCursor(2,1); //Set cursor to character 0 on line 1
707   switch(month){
708     case 1:
709       lcd.print("Month: January");
710       break;
711     case 2:
712       lcd.print("Month: February");
713       break;
714     case 3:
715       lcd.print("Month: March");
716       break;
717     case 4:
718       lcd.print("Month: April");
719       break;
720     case 5:
721       lcd.print("Month: Mai");
722       break;
723     case 6:
724       lcd.print("Month: June");
725       break;
726     case 7:
727       lcd.print("Month: July");
728       break;
729     case 8:
730       lcd.print("Month: August");
731       break;
732     case 9:
733       lcd.print("Month: September");
734       break;
735     case 10:
736       lcd.print("Month: October");
737       break;
738     case 11:
739       lcd.print("Month: November");
740       break;
741     case 0:
742       lcd.print("Month: December");
743       break;
744     case 12:
745       lcd.print("Month: All");
746       break;
747   }
748   print_min_max();
749 }
750
751 void print_mode(){
```

```
752   lcd.setCursor(0,0);
753   switch (mode) {
754   case 0:
755     lcd.print("1/5 TEMPERATURE");
756     break;
757   case 1:
758     lcd.print("2/5 GRID EMISSIONS");
759     break;
760   case 2:
761     lcd.print("3/5 IRRADIATION");
762     break;
763   case 3:
764     lcd.print("4/5 ENERGY DEMAND");
765     break;
766   case 4:
767     lcd.print("5/5 SUN POSITION");
768     break;
769
770   }
771   print_min_max();
772
773 }
774
775 void print_hour(){
776   lcd.setCursor(2,2);
777    switch(hour){
778         case 0:
779           lcd.print("Hour: 08:00");
780           break;
781         case 1:
782           lcd.print("Hour: 09:00");
783           break;
784         case 2:
785           lcd.print("Hour: 10:00");
786           break;
787         case 3:
788           lcd.print("Hour: 11:00");
789           break;
790         case 4:
791           lcd.print("Hour: 12:00");
792           break;
793         case 5:
794           lcd.print("Hour: 13:00");
795           break;
796         case 6:
797           lcd.print("Hour: 14:00");
798           break;
799         case 7:
800           lcd.print("Hour: 15:00");
801           break;
802         case 8:
803           lcd.print("Hour: 16:00");
804           break;
805         case 9:
806           lcd.print("Hour: 17:00");
807           break;
808         case 10:
809           lcd.print("Hour: 18:00");
810           break;
811         case 11:
812           lcd.print("Hour: ALL");
813           break;
814    }
815    print_min_max();
816 }
817
818 void print_min_max(){
819   clearLCDLine(3);
820   lcd.setCursor(1,3);
821   if(month==12 && hour==11){
822   lcd.setCursor(1,3);
```

```
823   switch (mode) {
824     case 0:
825     lcd.print("MIN:"+String(round(min_temp_si)));
826     lcd.setCursor(12, 3);
827     lcd.print("MAX:"+String(round(max_temp_si)));
828     break;
829     case 1:
830     lcd.print("MIN:"+String(round(min_grid_si)));
831     lcd.setCursor(12, 3);
832     lcd.print("MAX:"+String(round(max_grid_si)));
833     break;
834     case 2:
835     lcd.print("MIN:"+String(round(min_ghi_si)));
836     lcd.setCursor(12, 3);
837     lcd.print("MAX:"+String(round(max_ghi_si)));
838     break;
839   case 3:
840     lcd.print("MIN:"+String(round(min_dema_si)));
841     lcd.setCursor(12, 3);
842     lcd.print("MAX:"+String(round(max_dema_si)));
843     break;
844    case 4:
845     break;
846
847   }
848   }else if(month<12 && hour<11){
849     lcd.setCursor(1,3);
850     if(month==0){
851         aa=11;
852       }else{
853         aa=month-1;
854       }
855     switch (mode) {
856       case 0:
857         switch (hour) {
858         case 0:
859           lcd.print("Temp: "+String(round(avgtemp_8[aa])));
860           break;
861         case 1:
862           lcd.print("Temp: "+String(round(avgtemp_9[aa])));
863           break;
864         case 2:
865           lcd.print("Temp: "+String(round(avgtemp_10[aa])));
866           break;
867         case 3:
868           lcd.print("Temp: "+String(round(avgtemp_11[aa])));
869           break;
870         case 4:
871           lcd.print("Temp: "+String(round(avgtemp_12[aa])));
872           break;
873         case 5:
874           lcd.print("Temp: "+String(round(avgtemp_13[aa])));
875           break;
876         case 6:
877           lcd.print("Temp: "+String(round(avgtemp_14[aa])));
878           break;
879         case 7:
880           lcd.print("Temp: "+String(round(avgtemp_15[aa])));
881           break;
882         case 8:
883           lcd.print("Temp: "+String(round(avgtemp_16[aa])));
884           break;
885         case 9:
886           lcd.print("Temp: "+String(round(avgtemp_17[aa])));
887           break;
888         case 10:
889           lcd.print("Temp: "+String(round(avgtemp_18[aa])));
890           break;
891         }
892         break;
893       case 1:
```

```
894           switch (hour) {
895           case 0:
896             lcd.print("Emis: "+String(round(avggrid_8[aa]))+unit_grid);
897             break;
898           case 1:
899             lcd.print("Emis: "+String(round(avggrid_9[aa]))+unit_grid);
900             break;
901           case 2:
902             lcd.print("Emis: "+String(round(avggrid_10[aa]))+unit_grid);
903             break;
904           case 3:
905             lcd.print("Emis: "+String(round(avggrid_11[aa]))+unit_grid);
906             break;
907           case 4:
908             lcd.print("Emis: "+String(round(avggrid_12[aa]))+unit_grid);
909             break;
910           case 5:
911             lcd.print("Emis: "+String(round(avggrid_13[aa]))+unit_grid);
912             break;
913           case 6:
914             lcd.print("Emis: "+String(round(avggrid_14[aa]))+unit_grid);
915             break;
916           case 7:
917             lcd.print("Emis: "+String(round(avggrid_15[aa]))+unit_grid);
918             break;
919           case 8:
920             lcd.print("Emis: "+String(round(avggrid_16[aa]))+unit_grid);
921             break;
922           case 9:
923             lcd.print("Emis: "+String(round(avggrid_17[aa]))+unit_grid);
924             break;
925           case 10:
926             lcd.print("Emis: "+String(round(avggrid_18[aa]))+unit_grid);
927             break;
928           }
929           break;
930         case 2:
931           switch (hour) {
932           case 0:
933             lcd.print("Irrad: "+String(round(avgghi_8[aa]))+unit_ghi);
934             break;
935           case 1:
936             lcd.print("Irrad: "+String(round(avgghi_9[aa]))+unit_ghi);
937             break;
938           case 2:
939             lcd.print("Irrad: "+String(round(avgghi_10[aa]))+unit_ghi);
940             break;
941           case 3:
942             lcd.print("Irrad: "+String(round(avgghi_11[aa]))+unit_ghi);
943             break;
944           case 4:
945             lcd.print("Irrad: "+String(round(avgghi_12[aa]))+unit_ghi);
946             break;
947           case 5:
948             lcd.print("Irrad: "+String(round(avgghi_13[aa]))+unit_ghi);
949             break;
950           case 6:
951             lcd.print("Irrad: "+String(round(avgghi_14[aa]))+unit_ghi);
952             break;
953           case 7:
954             lcd.print("Irrad: "+String(round(avgghi_15[aa]))+unit_ghi);
955             break;
956           case 8:
957             lcd.print("Irrad: "+String(round(avgghi_16[aa]))+unit_ghi);
958             break;
959           case 9:
960             lcd.print("Irrad: "+String(round(avgghi_17[aa]))+unit_ghi);
961             break;
962           case 10:
963             lcd.print("Irrad: "+String(round(avgghi_18[aa]))+unit_ghi);
964             break;
```

```
965           }
966           break;
967         case 3:
968         switch (hour) {
969           case 0:
970             lcd.print("Demand: "+String(round(avgdema_8[aa]))+unit_dema);
971             break;
972           case 1:
973             lcd.print("Demand: "+String(round(avgdema_9[aa]))+unit_dema);
974             break;
975           case 2:
976             lcd.print("Demand: "+String(round(avgdema_10[aa]))+unit_dema);
977             break;
978           case 3:
979             lcd.print("Demand: "+String(round(avgdema_11[aa]))+unit_dema);
980             break;
981           case 4:
982             lcd.print("Demand: "+String(round(avgdema_12[aa]))+unit_dema);
983             break;
984           case 5:
985             lcd.print("Demand: "+String(round(avgdema_13[aa]))+unit_dema);
986             break;
987           case 6:
988             lcd.print("Demand: "+String(round(avgdema_14[aa]))+unit_dema);
989             break;
990           case 7:
991             lcd.print("Demand: "+String(round(avgdema_15[aa]))+unit_dema);
992             break;
993           case 8:
994             lcd.print("Demand: "+String(round(avgdema_16[aa]))+unit_dema);
995             break;
996           case 9:
997             lcd.print("Demand: "+String(round(avgdema_17[aa]))+unit_dema);
998             break;
999           case 10:
1000            lcd.print("Demand: "+String(round(avgdema_18[aa]))+unit_dema);
1001            break;
1002          }
1003          break;
1004        case 4:
1005          break;
1006    }
1007    }else if(month<12 && hour==11){
1008      if(month==0){
1009          aa=11;
1010        }else{
1011          aa=month-1;
1012        }
1013      switch (mode) {
1014        case 0:
1015          temp_array[0]=avgtemp_8[aa];
1016          temp_array[1]=avgtemp_9[aa];
1017          temp_array[2]=avgtemp_10[aa];
1018          temp_array[3]=avgtemp_11[aa];
1019          temp_array[4]=avgtemp_12[aa];
1020          temp_array[5]=avgtemp_13[aa];
1021          temp_array[6]=avgtemp_14[aa];
1022          temp_array[7]=avgtemp_15[aa];
1023          temp_array[8]=avgtemp_16[aa];
1024          temp_array[9]=avgtemp_17[aa];
1025          temp_array[10]=avgtemp_18[aa];
1026          break;
1027        case 1:
1028          temp_array[0]=avggrid_8[aa];
1029          temp_array[1]=avggrid_9[aa];
1030          temp_array[2]=avggrid_10[aa];
1031          temp_array[3]=avggrid_11[aa];
1032          temp_array[4]=avggrid_12[aa];
1033          temp_array[5]=avggrid_13[aa];
1034          temp_array[6]=avggrid_14[aa];
1035          temp_array[7]=avggrid_15[aa];
```

```
1036          temp_array[8]=avggrid_16[aa];
1037          temp_array[9]=avggrid_17[aa];
1038          temp_array[10]=avggrid_18[aa];
1039          break;
1040        case 2:
1041          temp_array[0]=avgghi_8[aa];
1042          temp_array[1]=avgghi_9[aa];
1043          temp_array[2]=avgghi_10[aa];
1044          temp_array[3]=avgghi_11[aa];
1045          temp_array[4]=avgghi_12[aa];
1046          temp_array[5]=avgghi_13[aa];
1047          temp_array[6]=avgghi_14[aa];
1048          temp_array[7]=avgghi_15[aa];
1049          temp_array[8]=avgghi_16[aa];
1050          temp_array[9]=avgghi_17[aa];
1051          temp_array[10]=avgghi_18[aa];
1052          break;
1053        case 3:
1054          temp_array[0]=avgdema_8[aa];
1055          temp_array[1]=avgdema_9[aa];
1056          temp_array[2]=avgdema_10[aa];
1057          temp_array[3]=avgdema_11[aa];
1058          temp_array[4]=avgdema_12[aa];
1059          temp_array[5]=avgdema_13[aa];
1060          temp_array[6]=avgdema_14[aa];
1061          temp_array[7]=avgdema_15[aa];
1062          temp_array[8]=avgdema_16[aa];
1063          temp_array[9]=avgdema_17[aa];
1064          temp_array[10]=avgdema_18[aa];
1065          break;
1066        case 4:
1067          break;
1068      }
1069      min_max(temp_array);
1070      lcd.print("MIN:"+String(round(minValue)));
1071      lcd.setCursor(12, 3);
1072      lcd.print("MAX:"+String(round(maxValue)));
1073
1074    }else if(month==12 && hour<11){
1075      if(month==0){
1076          aa=11;
1077        }else{
1078          aa=month-1;
1079        }
1080      lcd.setCursor(1,3);
1081        switch (mode) {
1082        case 0:
1083          switch (hour) {
1084          case 0:
1085            min_max(avgtemp_8);
1086            break;
1087          case 1:
1088            min_max(avgtemp_9);
1089            break;
1090          case 2:
1091            min_max(avgtemp_10);
1092            break;
1093          case 3:
1094            min_max(avgtemp_11);
1095            break;
1096          case 4:
1097            min_max(avgtemp_12);
1098            break;
1099          case 5:
1100            min_max(avgtemp_13);
1101            break;
1102          case 6:
1103            min_max(avgtemp_14);
1104            break;
1105          case 7:
1106            min_max(avgtemp_15);
```

```
1107            break;
1108          case 8:
1109            min_max(avgtemp_16);
1110            break;
1111          case 9:
1112            min_max(avgtemp_17);
1113            break;
1114          case 10:
1115            min_max(avgtemp_18);
1116            break;
1117          }
1118          lcd.print("MIN:"+String(round(minValue)));
1119          lcd.setCursor(12, 3);
1120          lcd.print("MAX:"+String(round(maxValue)));
1121          break;
1122        case 1:
1123          switch (hour) {
1124          case 0:
1125            min_max(avggrid_8);
1126            break;
1127          case 1:
1128            min_max(avggrid_9);
1129            break;
1130          case 2:
1131            min_max(avggrid_10);
1132            break;
1133          case 3:
1134            min_max(avggrid_11);
1135            break;
1136          case 4:
1137            min_max(avggrid_12);
1138            break;
1139          case 5:
1140            min_max(avggrid_13);
1141            break;
1142          case 6:
1143            min_max(avggrid_14);
1144            break;
1145          case 7:
1146            min_max(avggrid_15);
1147            break;
1148          case 8:
1149            min_max(avggrid_16);
1150            break;
1151          case 9:
1152            min_max(avggrid_17);
1153            break;
1154          case 10:
1155            min_max(avggrid_18);
1156            break;
1157          }
1158          lcd.print("MIN:"+String(round(minValue)));
1159          lcd.setCursor(12, 3);
1160          lcd.print("MAX:"+String(round(maxValue)));
1161          break;
1162        case 2:
1163          switch (hour) {
1164          case 0:
1165            min_max(avgghi_8);
1166            break;
1167          case 1:
1168            min_max(avgghi_9);
1169            break;
1170          case 2:
1171            min_max(avgghi_10);
1172            break;
1173          case 3:
1174            min_max(avgghi_11);
1175            break;
1176          case 4:
1177            min_max(avgghi_12);
```

```
1178            break;
1179          case 5:
1180            min_max(avgghi_13);
1181            break;
1182          case 6:
1183            min_max(avgghi_14);
1184            break;
1185          case 7:
1186            min_max(avgghi_15);
1187            break;
1188          case 8:
1189            min_max(avgghi_16);
1190            break;
1191          case 9:
1192            min_max(avgghi_17);
1193            break;
1194          case 10:
1195            min_max(avgghi_18);
1196            break;
1197          }
1198          lcd.print("MIN:"+String(round(minValue)));
1199          lcd.setCursor(12, 3);
1200          lcd.print("MAX:"+String(round(maxValue)));
1201          break;
1202
1203        case 3:
1204        switch (hour) {
1205          case 0:
1206            min_max(avgdema_8);
1207            break;
1208          case 1:
1209            min_max(avgdema_9);
1210            break;
1211          case 2:
1212            min_max(avgdema_10);
1213            break;
1214          case 3:
1215            min_max(avgdema_11);
1216            break;
1217          case 4:
1218            min_max(avgdema_12);
1219            break;
1220          case 5:
1221            min_max(avgdema_13);
1222            break;
1223          case 6:
1224            min_max(avgdema_14);
1225            break;
1226          case 7:
1227            min_max(avgdema_15);
1228            break;
1229          case 8:
1230            min_max(avgdema_16);
1231            break;
1232          case 9:
1233            min_max(avgdema_17);
1234            break;
1235          case 10:
1236            min_max(avgdema_18);
1237            break;
1238          }
1239          lcd.print("MIN:"+String(round(minValue)));
1240          lcd.setCursor(12, 3);
1241          lcd.print("MAX:"+String(round(maxValue)));
1242          break;
1243        case 4:
1244          break;
1245    }
1246
1247    }
1248 }
```

```
1249 void clearLCDLine(int line){
1250 for(int n = 0; n < 20; n++) {
1251 lcd.setCursor(n,line);
1252 lcd.print(" ");
1253 }
1254 }
1255
1256 void min_max(float ar[]){
1257   minValue = Min(ar[0], ar[1]);
1258   maxValue = Max(ar[0], ar[1]);
1259   for (int i = 2; i < 12; i++) {
1260     if(minValue>ar[a]){
1261       minValue=ar[a];
1262     }
1263     if (maxValue<ar[a]) {
1264       maxValue=ar[a];
1265     }
1266   }
1267 }
```

Annex B: Code for the Zurich sun path visualisation

```
#include <Adafruit_NeoPixel.h>
#include <PGMWrap.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27,20,4);

#define PIX_4 18
#define PIX_5 19
#define PIX_6 2
#define PIX_7 3
#define PIX_8 4
#define PIX_9 5
#define PIX_10 6
#define PIX_11 7
#define PIX_12 8
#define PIX_13 9
#define PIX_14 10
#define PIX_15 11
#define PIX_16 12
#define PIX_17 13
#define PIX_18 14
#define PIX_19 15


#define CONT_1 20
#define CONT_2 21
#define CONT_3 22

#define ENC_mode_A 24
#define ENC_mode_B 25
#define ENC_month_A 28
#define ENC_month_B 29
#define ENC_hour_A 46
#define ENC_hour_B 47

#define NUMPIXELS 12
#define DELAYVAL 100

//initialize pixles for the analemmas
Adafruit_NeoPixel pixels_4(NUMPIXELS, PIX_4, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_5(NUMPIXELS, PIX_5, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_6(NUMPIXELS, PIX_6, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_7(NUMPIXELS, PIX_7, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_8(NUMPIXELS, PIX_8, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_9(NUMPIXELS, PIX_9, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_10(NUMPIXELS, PIX_10, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_11(NUMPIXELS, PIX_11, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_12(NUMPIXELS, PIX_12, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_13(NUMPIXELS, PIX_13, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_14(NUMPIXELS, PIX_14, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_15(NUMPIXELS, PIX_15, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_16(NUMPIXELS, PIX_16, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_17(NUMPIXELS, PIX_17, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_18(NUMPIXELS, PIX_18, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel pixels_19(NUMPIXELS, PIX_19, NEO_GRBW + NEO_KHZ800);

//initialize pixles for the control panel
Adafruit_NeoPixel control_1(NUMPIXELS, CONT_1, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel control_2(NUMPIXELS, CONT_2, NEO_GRBW + NEO_KHZ800);
Adafruit_NeoPixel control_3(NUMPIXELS, CONT_3, NEO_GRBW + NEO_KHZ800);



String unit_temp="C";
String unit_grid="kgCO2/kWh";
String unit_ghi="W/m2";
String unit_dema="kWh";
float temp_array[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
volatile int mode=0;
```

```
volatile int month=12;
volatile int hour=16;
float minValue;
float maxValue;
int lower_b;
int upper_b;
int period=200;
int time_now;
int encoderPINA_mode = ENC_mode_A;
int encoderPINB_mode = ENC_mode_B;
int encoderPos_mode = 0;
int encoderPINALast_mode = LOW;
int encoderPINANow_mode = LOW;
int encoderPINA_month = ENC_month_A;
int encoderPINB_month = ENC_month_B;
int encoderPos_month = 0;
int encoderPINALast_month = LOW;
int encoderPINANow_month = LOW;
int encoderPINA_hour = ENC_hour_A;
int encoderPINB_hour = ENC_hour_B;
int encoderPos_hour = 0;
int encoderPINALast_hour = LOW;
int encoderPINANow_hour = LOW;
int a;
int aa;

// ZURICH GHI
float avgghi_4[]={0,0,0,0,0,1,0,0,0,0,0,0};
float avgghi_5[]={0,0,0,2,34,62,42,4,0,0,0,0};
float avgghi_6[]={0,0,2,57,128,161,144,96,21,1,0,0};
float avgghi_7[]={0,2,59,162,251,290,261,213,131,43,1,0};
float avgghi_8[]={7,58,164,289,361,399,386,340,244,135,44,6};
float avgghi_9[]={65,137,271,398,472,526,501,441,332,200,104,64};
float avgghi_10[]={113,208,345,495,522,596,580,517,407,249,147,111};
float avgghi_11[]={163,248,396,491,548,601,585,542,446,267,176,145};
float avgghi_12[]={170,276,429,526,592,567,564,554,444,280,182,134};
float avgghi_13[]={187,323,432,523,545,561,548,535,438,301,208,146};
float avgghi_14[]={145,264,385,469,489,519,532,478,389,253,153,108};
float avgghi_15[]={88,186,293,392,423,455,464,399,321,170,84,56};
float avgghi_16[]={19,100,191,295,331,353,355,312,215,79,9,2};
float avgghi_17[]={0,9,85,163,208,251,252,196,102,6,0,0};
float avgghi_18[]={0,0,3,48,101,134,139,89,7,0,0,0};
float avgghi_19[]={0,0,0,0,15,53,48,5,0,0,0,0};
float avgghi_20[]={0,0,0,0,0,1,0,0,0,0,0,0};
float g_ghi=200;
float max_ghi=728;
float min_ghi=1;
float max_ghi_si=601;
float min_ghi_si=1;

// ZURICH TEMP
float avgtemp_4[]={-1,0,2,5,9,12,14,14,10,7,3,0};
float avgtemp_5[]={-1,-1,2,5,9,12,14,14,10,7,2,0};
float avgtemp_6[]={-1,-1,1,5,10,13,14,14,9,7,2,0};
float avgtemp_7[]={-2,-1,1,5,11,14,15,15,10,7,2,0};
float avgtemp_8[]={-2,-1,2,7,12,16,17,16,11,8,2,0};
float avgtemp_9[]={-1,0,4,8,14,17,18,18,13,9,3,0};
float avgtemp_10[]={0,1,5,10,15,19,20,19,15,11,4,1};
float avgtemp_11[]={1,3,7,12,17,20,21,21,16,12,6,2};
float avgtemp_12[]={2,4,8,13,18,21,22,22,17,13,6,3};
float avgtemp_13[]={3,5,9,14,19,22,23,22,18,13,7,4};
float avgtemp_14[]={3,5,10,14,19,22,23,23,18,14,8,4};
float avgtemp_15[]={4,6,10,15,19,23,24,23,19,14,8,4};
float avgtemp_16[]={3,6,10,15,19,23,24,23,19,14,8,4};
float avgtemp_17[]={2,5,10,14,19,23,23,23,18,13,7,3};
float avgtemp_18[]={2,4,9,14,18,22,23,22,17,12,6,3};
float avgtemp_19[]={2,4,8,12,18,21,22,21,16,12,6,2};
float avgtemp_20[]={1,3,7,11,16,20,21,20,15,11,5,2};
float g_temp=0;
float max_temp=32;
float min_temp=-2;
```

```
141 float max_temp_si=24;
142 float min_temp_si=-2;
143
144 // ZURICH GRID EMISSIONS
145 float avggrid_4[]={125,149,137,102,41,49,64,105,125,166,144,139};
146 float avggrid_5[]={122,138,128,89,37,39,51,93,106,144,145,140};
147 float avggrid_6[]={110,121,113,81,36,37,49,84,95,128,133,135};
148 float avggrid_7[]={104,114,105,79,37,39,50,81,95,123,129,132};
149 float avggrid_8[]={102,109,101,81,40,44,55,81,99,121,127,128};
150 float avggrid_9[]={99,106,101,83,43,48,59,80,102,120,127,126};
151 float avggrid_10[]={98,107,103,84,44,50,62,84,103,119,127,126};
152 float avggrid_11[]={98,111,106,87,45,53,64,88,107,125,129,128};
153 float avggrid_12[]={103,118,110,89,46,56,67,92,111,133,133,131};
154 float avggrid_13[]={108,126,113,91,46,57,69,93,114,142,136,137};
155 float avggrid_14[]={112,133,116,94,46,56,69,95,116,147,138,139};
156 float avggrid_15[]={113,136,118,97,43,49,63,94,117,145,135,134};
157 float avggrid_16[]={102,125,120,96,38,41,55,86,109,127,122,126};
158 float avggrid_17[]={91,104,108,90,33,33,49,80,95,108,115,124};
159 float avggrid_18[]={91,96,94,86,31,30,47,80,91,111,119,126};
160 float avggrid_19[]={106,110,107,91,32,33,49,85,108,139,129,133};
161
162 float b_grid=0;
163 float max_grid=145;
164 float min_grid=30;
165 float max_grid_si=192;
166 float min_grid_si=88;
167
168 // ZURICH DEMAND
169 float avgdema_4[]={908,1067,746,370,123,118,118,113,141,263,689,1007};
170 float avgdema_5[]={908,1067,746,370,123,118,118,113,141,263,689,1007};
171 float avgdema_6[]={2711,2914,2461,1852,578,319,306,291,800,1599,2440,2838};
172 float avgdema_7[]={2850,3044,2593,1934,598,284,273,266,844,1740,2576,2960};
173 float avgdema_8[]={2847,3038,2550,1852,594,278,267,268,832,1731,2564,2950};
174 float avgdema_9[]={2744,2879,2386,1686,533,242,228,232,760,1603,2443,2828};
175 float avgdema_10[]={2626,2744,2244,1543,481,224,206,207,705,1487,2321,2717};
176 float avgdema_11[]={2502,2589,2094,1396,468,223,199,192,657,1375,2213,2605};
177 float avgdema_12[]={2158,2225,1718,1056,370,267,198,179,525,1048,1862,2266};
178 float avgdema_13[]={2284,2363,1836,1182,390,236,198,182,577,1174,1991,2405};
179 float avgdema_14[]={2203,2260,1735,1058,298,154,115,101,497,1109,1925,2327};
180 float avgdema_15[]={2154,2191,1660,977,262,133,96,79,465,1063,1877,2288};
181 float avgdema_16[]={2152,2192,1664,966,266,141,100,91,474,1088,1882,2275};
182 float avgdema_17[]={2182,2226,1705,993,344,232,184,172,540,1132,1914,2308};
183 float avgdema_18[]={2020,2098,1583,903,380,307,231,213,524,1020,1761,2147};
184 float avgdema_19[]={1924,2024,1521,862,366,298,223,193,490,956,1675,2041};
185 float avgdema_20[]={2122,2242,1765,1094,360,216,202,192,555,1180,1882,2250};
186 float g_dema=20;
187 float max_dema=3044;
188 float min_dema=-0;
189 float max_dema_si=3044;
190 float min_dema_si=79;
191
192
193 int r_sun=120;
194 int g_sun=120;
195 int b_sun=120;
196 int w_sun=255;
197
198 int reset_timeout=9000;
199 int counter=0;
200 int turn_flag=0;
201
202 void setup() {
203   watchdogSetup();
204   watchdogEnable(10000);
205   Serial.begin (115200);
206   //initialize LCD
207   lcd.init();
208   Wire.setClock(400000);
209   lcd.clear();
210   lcd.backlight();
211   //initialize Pixels
```

```
212   pixels_4.begin();
213   pixels_5.begin();
214   pixels_6.begin();
215   pixels_7.begin();
216   pixels_8.begin();
217   pixels_9.begin();
218   pixels_10.begin();
219   pixels_11.begin();
220   pixels_12.begin();
221   pixels_13.begin();
222   pixels_14.begin();
223   pixels_15.begin();
224   pixels_16.begin();
225   pixels_17.begin();
226   pixels_18.begin();
227   pixels_19.begin();
228   pinMode (encoderPINA_mode, INPUT_PULLUP);
229   pinMode (encoderPINB_mode, INPUT_PULLUP);
230   pinMode (encoderPINA_month, INPUT_PULLUP);
231   pinMode (encoderPINB_month, INPUT_PULLUP);
232   pinMode (encoderPINA_hour, INPUT_PULLUP);
233   pinMode (encoderPINB_hour, INPUT_PULLUP);
234   attachInterrupt(digitalPinToInterrupt(encoderPINA_month), turn_month, CHANGE);
235   attachInterrupt(digitalPinToInterrupt(encoderPINA_hour), turn_hour, CHANGE);
236   attachInterrupt(digitalPinToInterrupt(encoderPINA_mode), turn_mode, CHANGE);
237   randomSeed(analogRead(A3));
238   mode=random(0,5);
239   print_mode();
240   print_month();
241   print_hour();
242   // put your setup code here, to run once:
243
244 }
245
246
247 void loop() {
248   watchdogReset();
249   if (turn_flag==1){
250     counter=0;
251     turn_flag=0;
252   }
253   if( counter  >= reset_timeout && turn_flag==0 ){
254     rstc_start_software_reset(RSTC);
255   }
256   counter++;
257
258   switch (month) {
259   case 0:
260   case 1:
261   case 2:
262   case 3:
263   case 4:
264   case 5:
265   case 6:
266   case 7:
267   case 8:
268   case 9:
269   case 10:
270   case 11:
271   upper_b=month+1;
272   lower_b=month;
273   break;
274   case 12:
275   upper_b=12;
276   lower_b=0;
277   break;
278   }
279
280
281   //mode switch
282   switch (mode) {
```

```
283   case 0:
284   //clear_pix();
285   temp();
286   show();
287   break;
288   case 1:
289   //clear_pix();
290   grid();
291   show();
292   break;
293   case 2:
294   //clear_pix();
295   ghi();
296   show();
297   break;
298   case 3:
299   dema();
300   show();
301   //clear_pix();
302   break;
303   case 4:
304   sunpos();
305   show();
306   break;
307   }
308
309
310
311   turn_mode();
312   turn_month();
313   turn_hour();
314   //clear_pix();
315
316
317 }
318   void ghi(){
319
320     for (int i = lower_b; i < upper_b; i++) {
321       if(i==0){
322         a=11;
323       }else{
324         a=i-1;
325       }
326       switch(hour){
327         case 0:
328           pixels_4.setPixelColor(i,120+125*(avgghi_4[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_4[a])/(max_ghi-min_ghi),0);
329           break;
330         case 1:
331           pixels_5.setPixelColor(i,120+125*(avgghi_5[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_5[a])/(max_ghi-min_ghi),0);
332           break;
333         case 2:
334         if(i>=4){
335           pixels_6.setPixelColor(i-4,120+125*(avgghi_6[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_6[a])/(max_ghi-min_ghi),0);
336           }
337           break;
338         case 3:
339         if(i>=3){
340           pixels_7.setPixelColor(i-3,120+125*(avgghi_7[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_7[a])/(max_ghi-min_ghi),0);
341           }
342           break;
343         case 4:
344         if(i>=2){
345           pixels_8.setPixelColor(i-2,120+125*(avgghi_8[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_8[a])/(max_ghi-min_ghi),0);
346           }
347           break;
348         case 5:
```

```
349           pixels_9.setPixelColor(i,120+125*(avgghi_9[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_9[a])/(max_ghi-min_ghi),0);
350           break;
351         case 6:
352           pixels_10.setPixelColor(i,120+125*(avgghi_10[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_10[a])/(max_ghi-min_ghi),0);
353           break;
354         case 7:
355           pixels_11.setPixelColor(i,120+125*(avgghi_11[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_11[a])/(max_ghi-min_ghi),0);
356           break;
357         case 8:
358           pixels_12.setPixelColor(i,120+125*(avgghi_12[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_12[a])/(max_ghi-min_ghi),0);
359           break;
360         case 9:
361           pixels_13.setPixelColor(i,120+125*(avgghi_13[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_13[a])/(max_ghi-min_ghi),0);
362           break;
363         case 10:
364           pixels_14.setPixelColor(i,120+125*(avgghi_14[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_14[a])/(max_ghi-min_ghi),0);
365           break;
366         case 11:
367           pixels_15.setPixelColor(i,120+125*(avgghi_15[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_15[a])/(max_ghi-min_ghi),0);
368           break;
369         case 12:
370           pixels_16.setPixelColor(i,120+125*(avgghi_16[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_16[a])/(max_ghi-min_ghi),0);
371           break;
372         case 13:
373         if(i>=1){
374           pixels_17.setPixelColor(i-1,120+125*(avgghi_17[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,70*(max_ghi-avgghi_17[a])/(max_ghi-min_ghi),0);
375         }
376           break;
377         case 14:
378         if(i>=2){
379           pixels_18.setPixelColor(i-2,120+125*(avgghi_18[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,70*(max_ghi-avgghi_18[a])/(max_ghi-min_ghi),0);
380         }
381           break;
382         case 15:
383         if(i>=4){
384           pixels_19.setPixelColor(i-4,120+125*(avgghi_19[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
      ,70*(max_ghi-avgghi_19[a])/(max_ghi-min_ghi),0);
385         }
386           break;
387         case 16:
388           pixels_4.setPixelColor(i,120+125*(avgghi_4[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_4[a])/(max_ghi-min_ghi),0);
389           pixels_5.setPixelColor(i,120+125*(avgghi_5[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_5[a])/(max_ghi-min_ghi),0);
390           if(i>=4){
391           pixels_6.setPixelColor(i-4,120+125*(avgghi_6[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_6[a])/(max_ghi-min_ghi),0);
392           }
393           if(i>=3){
394           pixels_7.setPixelColor(i-3,120+125*(avgghi_7[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_7[a])/(max_ghi-min_ghi),0);
395           }
396           if(i>=2){
397           pixels_8.setPixelColor(i-2,120+125*(avgghi_8[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_8[a])/(max_ghi-min_ghi),0);
398           }
399           pixels_9.setPixelColor(i,120+125*(avgghi_9[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_9[a])/(max_ghi-min_ghi),0);
400           pixels_10.setPixelColor(i,120+125*(avgghi_10[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
      max_ghi-avgghi_10[a])/(max_ghi-min_ghi),0);
401           pixels_11.setPixelColor(i,120+125*(avgghi_11[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
```

```
    max_ghi-avgghi_11[a])/(max_ghi-min_ghi),0);
402         pixels_12.setPixelColor(i,120+125*(avgghi_12[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_12[a])/(max_ghi-min_ghi),0);
403         pixels_13.setPixelColor(i,120+125*(avgghi_13[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_13[a])/(max_ghi-min_ghi),0);
404         pixels_14.setPixelColor(i,120+125*(avgghi_14[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_14[a])/(max_ghi-min_ghi),0);
405         pixels_15.setPixelColor(i,120+125*(avgghi_15[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_15[a])/(max_ghi-min_ghi),0);
406         pixels_16.setPixelColor(i,120+125*(avgghi_16[a]-min_ghi)/(max_ghi-min_ghi), g_ghi,70*(
    max_ghi-avgghi_16[a])/(max_ghi-min_ghi),0);
407         if(i>=1){
408         pixels_17.setPixelColor(i-1,120+125*(avgghi_17[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,70*(max_ghi-avgghi_17[a])/(max_ghi-min_ghi),0);
409         }
410         if(i>=2){
411         pixels_18.setPixelColor(i-2,120+125*(avgghi_18[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,70*(max_ghi-avgghi_18[a])/(max_ghi-min_ghi),0);
412         }
413         if(i>=4){
414         pixels_19.setPixelColor(i-4,120+125*(avgghi_19[a]-min_ghi)/(max_ghi-min_ghi), g_ghi
    ,70*(max_ghi-avgghi_19[a])/(max_ghi-min_ghi),0);
415         }
416         break;
417   }
418   }
419 }
420
421 void temp(){
422   for (int i = lower_b; i < upper_b; i++) {
423     if(i==0){
424       a=11;
425     }else{
426       a=i-1;
427     }
428     switch(hour){
429       case 0:
430         pixels_4.setPixelColor(i,255*(avgtemp_4[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_4[a])/(max_temp-min_temp),0);
431         break;
432       case 1:
433         pixels_5.setPixelColor(i,255*(avgtemp_5[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_5[a])/(max_temp-min_temp),0);
434         break;
435       case 2:
436       if(i>=4){
437         pixels_6.setPixelColor(i-4,255*(avgtemp_6[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_6[a])/(max_temp-min_temp),0);
438         break;
439       }
440       case 3:
441       if(i>=3){
442         pixels_7.setPixelColor(i-3,255*(avgtemp_7[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_7[a])/(max_temp-min_temp),0);
443         break;
444       }
445       case 4:
446       if(i>=2){
447         pixels_8.setPixelColor(i-2,255*(avgtemp_8[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_8[a])/(max_temp-min_temp),0);
448         break;
449       }
450       case 5:
451         pixels_9.setPixelColor(i,255*(avgtemp_9[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_9[a])/(max_temp-min_temp),0);
452         break;
453       case 6:
454         pixels_10.setPixelColor(i,255*(avgtemp_10[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_10[a])/(max_temp-min_temp),0);
455         break;
456       case 7:
```

```
457             pixels_11.setPixelColor(i,255*(avgtemp_11[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_11[a])/(max_temp-min_temp),0);
458             break;
459         case 8:
460             pixels_12.setPixelColor(i,255*(avgtemp_12[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_12[a])/(max_temp-min_temp),0);
461             break;
462         case 9:
463             pixels_13.setPixelColor(i,255*(avgtemp_13[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_13[a])/(max_temp-min_temp),0);
464             break;
465         case 10:
466             pixels_14.setPixelColor(i,255*(avgtemp_14[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_14[a])/(max_temp-min_temp),0);
467             break;
468         case 11:
469             pixels_15.setPixelColor(i,255*(avgtemp_15[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_15[a])/(max_temp-min_temp),0);
470             break;
471         case 12:
472             pixels_16.setPixelColor(i,255*(avgtemp_16[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_16[a])/(max_temp-min_temp),0);
473             break;
474         case 13:
475         if(i>=1){
476             pixels_17.setPixelColor(i-1,255*(avgtemp_17[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_17[a])/(max_temp-min_temp),0);
477         }
478         break;
479         case 14:
480         if(i>=2){
481             pixels_18.setPixelColor(i-2,255*(avgtemp_18[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_18[a])/(max_temp-min_temp),0);
482         }
483             break;
484         case 15:
485         if(i>=4){
486             pixels_19.setPixelColor(i-4,255*(avgtemp_19[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_19[a])/(max_temp-min_temp),0);
487             }
488             break;
489         case 16:
490             pixels_4.setPixelColor(i,255*(avgtemp_4[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_4[a])/(max_temp-min_temp),0);
491             pixels_5.setPixelColor(i,255*(avgtemp_5[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_5[a])/(max_temp-min_temp),0);
492             if(i>=4){
493             pixels_6.setPixelColor(i-4,255*(avgtemp_6[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_6[a])/(max_temp-min_temp),0);
494             }
495             if(i>=3){
496             pixels_7.setPixelColor(i-3,255*(avgtemp_7[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_7[a])/(max_temp-min_temp),0);
497             }
498             if(i>=2){
499             pixels_8.setPixelColor(i-2,255*(avgtemp_8[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_8[a])/(max_temp-min_temp),0);
500             }
501             pixels_9.setPixelColor(i,255*(avgtemp_9[a]-min_temp)/(max_temp-min_temp), g_temp,255*(
    max_temp-avgtemp_9[a])/(max_temp-min_temp),0);
502             pixels_10.setPixelColor(i,255*(avgtemp_10[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_10[a])/(max_temp-min_temp),0);
503             pixels_11.setPixelColor(i,255*(avgtemp_11[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_11[a])/(max_temp-min_temp),0);
504             pixels_12.setPixelColor(i,255*(avgtemp_12[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_12[a])/(max_temp-min_temp),0);
505             pixels_13.setPixelColor(i,255*(avgtemp_13[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_13[a])/(max_temp-min_temp),0);
506             pixels_14.setPixelColor(i,255*(avgtemp_14[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_14[a])/(max_temp-min_temp),0);
507             pixels_15.setPixelColor(i,255*(avgtemp_15[a]-min_temp)/(max_temp-min_temp), g_temp
```

```
    ,255*(max_temp-avgtemp_15[a])/(max_temp-min_temp),0);
508         pixels_16.setPixelColor(i,255*(avgtemp_16[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_16[a])/(max_temp-min_temp),0);
509         if(i>=1){
510         pixels_17.setPixelColor(i-1,255*(avgtemp_17[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_17[a])/(max_temp-min_temp),0);
511         }
512         if(i>=2){
513         pixels_18.setPixelColor(i-2,255*(avgtemp_18[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_18[a])/(max_temp-min_temp),0);
514         }
515         if(i>=4){
516         pixels_19.setPixelColor(i-4,255*(avgtemp_19[a]-min_temp)/(max_temp-min_temp), g_temp
    ,255*(max_temp-avgtemp_19[a])/(max_temp-min_temp),0);
517         }
518         break;
519       }
520     }
521   }
522
523   void dema(){
524     for (int i = lower_b; i < upper_b; i++) {
525       if(i==0){
526         a=11;
527       }else{
528         a=i-1;
529       }
530       switch(hour){
531         case 0:
532           pixels_4.setPixelColor(i,115+140*(avgdema_4[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_4[a])/(max_dema-min_dema),0);
533           break;
534         case 1:
535           pixels_5.setPixelColor(i,115+140*(avgdema_5[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_5[a])/(max_dema-min_dema),0);
536           break;
537         case 2:
538         if(i>=4){
539           pixels_6.setPixelColor(i-4,115+140*(avgdema_6[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_6[a])/(max_dema-min_dema),0);
540         }
541           break;
542         case 3:
543         if(i>=3){
544           pixels_7.setPixelColor(i-3,115+140*(avgdema_7[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_7[a])/(max_dema-min_dema),0);
545         }
546           break;
547         case 4:
548         if(i>=2){
549           pixels_8.setPixelColor(i-2,115+140*(avgdema_8[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_8[a])/(max_dema-min_dema),0);
550         }
551           break;
552         case 5:
553           pixels_9.setPixelColor(i,115+140*(avgdema_9[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_9[a])/(max_dema-min_dema),0);
554           break;
555         case 6:
556           pixels_10.setPixelColor(i,115+140*(avgdema_10[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_10[a])/(max_dema-min_dema),0);
557           break;
558         case 7:
559           pixels_11.setPixelColor(i,115+140*(avgdema_11[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_11[a])/(max_dema-min_dema),0);
560           break;
561         case 8:
562           pixels_12.setPixelColor(i,115+140*(avgdema_12[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_12[a])/(max_dema-min_dema),0);
563           break;
564         case 9:
```

```
565             pixels_13.setPixelColor(i,115+140*(avgdema_13[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_13[a])/(max_dema-min_dema),0);
566             break;
567         case 10:
568             pixels_14.setPixelColor(i,115+140*(avgdema_14[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_14[a])/(max_dema-min_dema),0);
569             break;
570         case 11:
571             pixels_15.setPixelColor(i,115+140*(avgdema_15[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_15[a])/(max_dema-min_dema),0);
572             break;
573         case 12:
574             pixels_16.setPixelColor(i,115+140*(avgdema_16[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_16[a])/(max_dema-min_dema),0);
575             break;
576         case 13:
577         if(i>=1){
578             pixels_17.setPixelColor(i-1,115+140*(avgdema_17[a]-min_dema)/(max_dema-min_dema),
    g_dema,180*(max_dema-avgdema_17[a])/(max_dema-min_dema),0);
579         }
580         break;
581         case 14:
582         if(i>=2){
583             pixels_18.setPixelColor(i-2,115+140*(avgdema_18[a]-min_dema)/(max_dema-min_dema),
    g_dema,180*(max_dema-avgdema_18[a])/(max_dema-min_dema),0);
584         }
585         break;
586         case 15:
587         if(i>=4){
588             pixels_19.setPixelColor(i-4,115+140*(avgdema_19[a]-min_dema)/(max_dema-min_dema),
    g_dema,180*(max_dema-avgdema_19[a])/(max_dema-min_dema),0);
589         }
590         break;
591         case 16:
592             pixels_4.setPixelColor(i,115+140*(avgdema_4[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_4[a])/(max_dema-min_dema),0);
593             pixels_5.setPixelColor(i,115+140*(avgdema_5[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_5[a])/(max_dema-min_dema),0);
594             if(i>=4){
595             pixels_6.setPixelColor(i-4,115+140*(avgdema_6[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_6[a])/(max_dema-min_dema),0);
596             }
597             if(i>=3){
598             pixels_7.setPixelColor(i-3,115+140*(avgdema_7[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_7[a])/(max_dema-min_dema),0);
599             }
600             if(i>=2){
601             pixels_8.setPixelColor(i-2,115+140*(avgdema_8[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_8[a])/(max_dema-min_dema),0);
602             }
603             pixels_9.setPixelColor(i,115+140*(avgdema_9[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_9[a])/(max_dema-min_dema),0);
604             pixels_10.setPixelColor(i,115+140*(avgdema_10[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_10[a])/(max_dema-min_dema),0);
605             pixels_11.setPixelColor(i,115+140*(avgdema_11[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_11[a])/(max_dema-min_dema),0);
606             pixels_12.setPixelColor(i,115+140*(avgdema_12[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_12[a])/(max_dema-min_dema),0);
607             pixels_13.setPixelColor(i,115+140*(avgdema_13[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_13[a])/(max_dema-min_dema),0);
608             pixels_14.setPixelColor(i,115+140*(avgdema_14[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_14[a])/(max_dema-min_dema),0);
609             pixels_15.setPixelColor(i,115+140*(avgdema_15[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_15[a])/(max_dema-min_dema),0);
610             pixels_16.setPixelColor(i,115+140*(avgdema_16[a]-min_dema)/(max_dema-min_dema), g_dema
    ,180*(max_dema-avgdema_16[a])/(max_dema-min_dema),0);
611             if(i>=1){
612             pixels_17.setPixelColor(i-1,115+140*(avgdema_17[a]-min_dema)/(max_dema-min_dema),
    g_dema,180*(max_dema-avgdema_17[a])/(max_dema-min_dema),0);
613             }
614             if(i>=2){
```

```
615             pixels_18.setPixelColor(i-2,115+140*(avgdema_18[a]-min_dema)/(max_dema-min_dema),
    g_dema,180*(max_dema-avgdema_18[a])/(max_dema-min_dema),0);
616             }
617             if(i>=4){
618             pixels_19.setPixelColor(i-4,115+140*(avgdema_19[a]-min_dema)/(max_dema-min_dema),
    g_dema,180*(max_dema-avgdema_19[a])/(max_dema-min_dema),0);
619             }
620             break;
621         }
622     }
623   }
624
625 void grid(){
626     for (int i = lower_b; i < upper_b; i++) {
627       if(i==0){
628         a=11;
629       }else{
630         a=i-1;
631       }
632         switch (hour) {
633         case 0:
634           pixels_4.setPixelColor(i,255*(avggrid_4[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_4[a])/(max_grid-min_grid),b_grid,0);
635           break;
636         case 1:
637           pixels_5.setPixelColor(i,255*(avggrid_5[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_5[a])/(max_grid-min_grid),b_grid,0);
638           break;
639         case 2:
640         if(i>=4){
641           pixels_6.setPixelColor(i-4,255*(avggrid_6[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_6[a])/(max_grid-min_grid),b_grid,0);
642         }
643         break;
644         case 3:
645         if(i>=3){
646           pixels_7.setPixelColor(i-3,255*(avggrid_7[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_7[a])/(max_grid-min_grid),b_grid,0);
647         }
648         break;
649         case 4:
650         if(i>=2){
651           pixels_8.setPixelColor(i-2,255*(avggrid_8[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_8[a])/(max_grid-min_grid),b_grid,0);
652         }
653         break;
654         case 5:
655           pixels_9.setPixelColor(i,255*(avggrid_9[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_9[a])/(max_grid-min_grid),b_grid,0);
656           break;
657         case 6:
658           pixels_10.setPixelColor(i,255*(avggrid_10[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_10[a])/(max_grid-min_grid),b_grid,0);
659           break;
660         case 7:
661           pixels_11.setPixelColor(i,255*(avggrid_11[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_11[a])/(max_grid-min_grid),b_grid,0);
662           break;
663         case 8:
664           pixels_12.setPixelColor(i,255*(avggrid_12[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_12[a])/(max_grid-min_grid),b_grid,0);
665           break;
666         case 9:
667           pixels_13.setPixelColor(i,255*(avggrid_13[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_13[a])/(max_grid-min_grid),b_grid,0);
668           break;
669         case 10:
670           pixels_14.setPixelColor(i,255*(avggrid_14[a]-min_grid)/(max_grid-min_grid), 255*(
    max_grid-avggrid_14[a])/(max_grid-min_grid),b_grid,0);
671           break;
672         case 11:
```

```
673           pixels_15.setPixelColor(i,255*(avggrid_15[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_15[a])/(max_grid-min_grid),b_grid,0);
674           break;
675         case 12:
676           pixels_16.setPixelColor(i,255*(avggrid_16[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_16[a])/(max_grid-min_grid),b_grid,0);
677           break;
678         case 13:
679         if(i>=1){
680           pixels_17.setPixelColor(i-1,255*(avggrid_17[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_17[a])/(max_grid-min_grid),b_grid,0);
681         }
682         break;
683         case 14:
684         if(i>=2){
685           pixels_18.setPixelColor(i-2,255*(avggrid_18[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_18[a])/(max_grid-min_grid),b_grid,0);
686         }
687         break;
688         case 15:
689         if(i>=4){
690           pixels_19.setPixelColor(i-4,255*(avggrid_19[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_19[a])/(max_grid-min_grid),b_grid,0);
691         }
692         break;
693         case 16:
694           pixels_4.setPixelColor(i,255*(avggrid_4[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_4[a])/(max_grid-min_grid),b_grid,0);
695           pixels_5.setPixelColor(i,255*(avggrid_5[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_5[a])/(max_grid-min_grid),b_grid,0);
696           if(i>=4){
697           pixels_6.setPixelColor(i-4,255*(avggrid_6[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_6[a])/(max_grid-min_grid),b_grid,0);
698           }
699           if(i>=3){
700           pixels_7.setPixelColor(i-3,255*(avggrid_7[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_7[a])/(max_grid-min_grid),b_grid,0);
701           }
702           if(i>=2){
703           pixels_8.setPixelColor(i-2,255*(avggrid_8[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_8[a])/(max_grid-min_grid),b_grid,0);
704           }
705           pixels_9.setPixelColor(i,255*(avggrid_9[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_9[a])/(max_grid-min_grid),b_grid,0);
706           pixels_10.setPixelColor(i,255*(avggrid_10[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_10[a])/(max_grid-min_grid),b_grid,0);
707           pixels_11.setPixelColor(i,255*(avggrid_11[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_11[a])/(max_grid-min_grid),b_grid,0);
708           pixels_12.setPixelColor(i,255*(avggrid_12[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_12[a])/(max_grid-min_grid),b_grid,0);
709           pixels_13.setPixelColor(i,255*(avggrid_13[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_13[a])/(max_grid-min_grid),b_grid,0);
710           pixels_14.setPixelColor(i,255*(avggrid_14[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_14[a])/(max_grid-min_grid),b_grid,0);
711           pixels_15.setPixelColor(i,255*(avggrid_15[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_15[a])/(max_grid-min_grid),b_grid,0);
712           pixels_16.setPixelColor(i,255*(avggrid_16[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_16[a])/(max_grid-min_grid),b_grid,0);
713           if(i>=1){
714           pixels_17.setPixelColor(i-1,255*(avggrid_17[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_17[a])/(max_grid-min_grid),b_grid,0);
715           }
716           if(i>=2){
717           pixels_18.setPixelColor(i-2,255*(avggrid_18[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_18[a])/(max_grid-min_grid),b_grid,0);
718           }
719           if(i>=4){
720           pixels_19.setPixelColor(i-4,255*(avggrid_19[a]-min_grid)/(max_grid-min_grid), 255*(
      max_grid-avggrid_19[a])/(max_grid-min_grid),b_grid,0);
721           }
722           break;
```

```
723
724
725         }
726
727     }
728   }
729   void sunpos(){
730     for (int i = lower_b; i < upper_b; i++) {
731       switch (hour) {
732         case 0:
733           pixels_4.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
734           break;
735         case 1:
736           pixels_5.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
737           break;
738         case 2:
739         if(i>=4){
740           pixels_6.setPixelColor(i-4,r_sun,g_sun,b_sun,w_sun);
741         }
742           break;
743         case 3:
744         if(i>=3){
745           pixels_7.setPixelColor(i-3,r_sun,g_sun,b_sun,w_sun);
746         }
747           break;
748         case 4:
749         if(i>=2){
750           pixels_8.setPixelColor(i-2,r_sun,g_sun,b_sun,w_sun);
751         }
752           break;
753         case 5:
754           pixels_9.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
755           break;
756         case 6:
757           pixels_10.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
758           break;
759         case 7:
760           pixels_11.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
761           break;
762         case 8:
763           pixels_12.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
764           break;
765         case 9:
766           pixels_13.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
767           break;
768         case 10:
769           pixels_14.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
770           break;
771         case 11:
772           pixels_15.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
773           break;
774         case 12:
775           pixels_16.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
776           break;
777         case 13:
778         if(i>=1){
779           pixels_17.setPixelColor(i-1,r_sun,g_sun,b_sun,w_sun);
780         }
781           break;
782         case 14:
783         if(i>=2){
784           pixels_18.setPixelColor(i-2,r_sun,g_sun,b_sun,w_sun);
785         }
786           break;
787         case 15:
788         if(i>=4){
789           pixels_19.setPixelColor(i-4,r_sun,g_sun,b_sun,w_sun);
790         }
791           break;
792         case 16:
793
```

```
794             pixels_4.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
795             pixels_5.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
796             if(i>=4){
797             pixels_6.setPixelColor(i-4,r_sun,g_sun,b_sun,w_sun);
798             }
799             if(i>=3){
800             pixels_7.setPixelColor(i-3,r_sun,g_sun,b_sun,w_sun);
801             }
802             if(i>=2){
803             pixels_8.setPixelColor(i-2,r_sun,g_sun,b_sun,w_sun);
804             }
805             pixels_9.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
806             pixels_10.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
807             pixels_11.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
808             pixels_12.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
809             pixels_13.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
810             pixels_14.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
811             pixels_15.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
812             pixels_16.setPixelColor(i,r_sun,g_sun,b_sun,w_sun);
813             if(i>=1){
814             pixels_17.setPixelColor(i-1,r_sun,g_sun,b_sun,w_sun);
815             }
816             if(i>=2){
817             pixels_18.setPixelColor(i-2,r_sun,g_sun,b_sun,w_sun);
818             }
819             if(i>=4){
820             pixels_19.setPixelColor(i-4,r_sun,g_sun,b_sun,w_sun);
821             }
822             break;
823         }
824
825
826
827
828
829       }
830     }
831
832     void show(){
833       pixels_4.show();
834       pixels_5.show();
835       pixels_6.show();
836       pixels_7.show();
837       pixels_8.show();
838       pixels_9.show();
839       pixels_10.show();
840       pixels_11.show();
841       pixels_12.show();
842       pixels_13.show();
843       pixels_14.show();
844       pixels_15.show();
845       pixels_16.show();
846       pixels_17.show();
847       pixels_18.show();
848       pixels_19.show();
849     }
850
851   void clear_pix(){
852       pixels_4.clear();
853       pixels_5.clear();
854       pixels_6.clear();
855       pixels_7.clear();
856       pixels_8.clear();
857       pixels_9.clear();
858       pixels_10.clear();
859       pixels_11.clear();
860       pixels_12.clear();
861       pixels_13.clear();
862       pixels_14.clear();
863       pixels_15.clear();
864       pixels_16.clear();
```

```
865     pixels_17.clear();
866     pixels_18.clear();
867     pixels_19.clear();
868     show();
869 }
870
871 void turn_mode(){
872   encoderPINANow_mode = digitalRead(encoderPINA_mode);
873   if ((encoderPINALast_mode == HIGH) && (encoderPINANow_mode == LOW)) {
874     if (digitalRead(encoderPINB_mode) == HIGH) {
875       if (encoderPos_mode<4) {
876         encoderPos_mode++;
877       } else {
878       encoderPos_mode=0;
879       }
880     } else {
881       if(encoderPos_mode>0){
882         encoderPos_mode--;
883       } else if (encoderPos_mode==0) {
884         encoderPos_mode=4;
885       }
886     }
887
888     clear_pix();
889     clearLCDLine(0);
890     turn_flag=1;
891     mode=encoderPos_mode;
892     print_mode();
893   }
894   encoderPINALast_mode = encoderPINANow_mode;
895 }
896
897 void turn_month(){
898   encoderPINANow_month = digitalRead(encoderPINA_month);
899   if ((encoderPINALast_month == HIGH) && (encoderPINANow_month == LOW)) {
900     if (digitalRead(encoderPINB_month) == HIGH) {
901       if (encoderPos_month<12) {
902         encoderPos_month++;
903       } else {
904       encoderPos_month=0;
905       }
906     } else {
907       if(encoderPos_month>0){
908         encoderPos_month--;
909       } else if (encoderPos_month==0) {
910         encoderPos_month=12;
911       }
912     }
913
914     clear_pix();
915     clearLCDLine(1);
916     turn_flag=1;
917     month=encoderPos_month;
918     print_month();
919   }
920   encoderPINALast_month = encoderPINANow_month;
921 }
922
923 void turn_hour(){
924   encoderPINANow_hour = digitalRead(encoderPINA_hour);
925   if ((encoderPINALast_hour == HIGH) && (encoderPINANow_hour == LOW)) {
926     if (digitalRead(encoderPINB_hour) == HIGH) {
927       if (encoderPos_hour<16) {
928         encoderPos_hour++;
929       } else {
930       encoderPos_hour=2;
931       }
932     } else {
933       if(encoderPos_hour>0){
934         encoderPos_hour--;
935       } else if (encoderPos_hour<=2) {
```

```
936           encoderPos_hour=16;
937         }
938       }
939       clear_pix();
940       clearLCDLine(2);
941       turn_flag=1;
942       Serial.println(encoderPos_hour);
943       hour=encoderPos_hour;
944       print_hour();
945     }
946     encoderPINALast_hour = encoderPINANow_hour;
947   }
948
949   void print_month(){
950     lcd.setCursor(2,1); //Set cursor to character 0 on line 1
951     switch(month){
952       case 1:
953         lcd.print("Month: January");
954         break;
955       case 2:
956         lcd.print("Month: February");
957         break;
958       case 3:
959         lcd.print("Month: March");
960         break;
961       case 4:
962         lcd.print("Month: April");
963         break;
964       case 5:
965         lcd.print("Month: Mai");
966         break;
967       case 6:
968         lcd.print("Month: June");
969         break;
970       case 7:
971         lcd.print("Month: July");
972         break;
973       case 8:
974         lcd.print("Month: August");
975         break;
976       case 9:
977         lcd.print("Month: September");
978         break;
979       case 10:
980         lcd.print("Month: October");
981         break;
982       case 11:
983         lcd.print("Month: November");
984         break;
985       case 0:
986         lcd.print("Month: December");
987         break;
988       case 12:
989         lcd.print("Month: All");
990         break;
991     }
992     print_min_max();
993   }
994
995   void print_mode(){
996     lcd.setCursor(0,0);
997     switch (mode) {
998     case 0:
999       lcd.print("1/5 TEMPERATURE");
1000      break;
1001    case 1:
1002      lcd.print("2/5 GRID EMISSIONS");
1003      break;
1004    case 2:
1005      lcd.print("3/5 IRRADIATION");
1006      break;
```

```
1007    case 3:
1008      lcd.print("4/5 ENERGY DEMAND");
1009      break;
1010    case 4:
1011      lcd.print("5/5 SUN POSITION");
1012      break;
1013
1014    }
1015    print_min_max();
1016
1017  }
1018
1019  void print_hour(){
1020    lcd.setCursor(2,2);
1021     switch(hour){
1022          case 0:
1023            lcd.print("Hour: 04:00");
1024            break;
1025          case 1:
1026            lcd.print("Hour: 05:00");
1027            break;
1028          case 2:
1029            lcd.print("Hour: 06:00");
1030            break;
1031          case 3:
1032            lcd.print("Hour: 07:00");
1033            break;
1034          case 4:
1035            lcd.print("Hour: 08:00");
1036            break;
1037          case 5:
1038            lcd.print("Hour: 09:00");
1039            break;
1040          case 6:
1041            lcd.print("Hour: 10:00");
1042            break;
1043          case 7:
1044            lcd.print("Hour: 11:00");
1045            break;
1046          case 8:
1047            lcd.print("Hour: 12:00");
1048            break;
1049          case 9:
1050            lcd.print("Hour: 13:00");
1051            break;
1052          case 10:
1053            lcd.print("Hour: 14:00");
1054            break;
1055          case 11:
1056            lcd.print("Hour: 15:00");
1057            break;
1058          case 12:
1059            lcd.print("Hour: 16:00");
1060            break;
1061          case 13:
1062            lcd.print("Hour: 17:00");
1063            break;
1064          case 14:
1065            lcd.print("Hour: 18:00");
1066            break;
1067          case 15:
1068            lcd.print("Hour: 19:00");
1069            break;
1070          case 16:
1071            lcd.print("Hour: ALL");
1072            break;
1073     }
1074     print_min_max();
1075  }
1076
1077  void print_min_max(){
```

```
1078    clearLCDLine(3);
1079    lcd.setCursor(1,3);
1080    if(month==12 && hour==16){
1081    lcd.setCursor(1,3);
1082    switch (mode) {
1083      case 0:
1084      lcd.print("MIN:"+String(round(min_temp_si)));
1085      lcd.setCursor(12, 3);
1086      lcd.print("MAX:"+String(round(max_temp_si)));
1087      break;
1088      case 1:
1089      lcd.print("MIN:"+String(round(min_grid_si)));
1090      lcd.setCursor(12, 3);
1091      lcd.print("MAX:"+String(round(max_grid_si)));
1092      break;
1093      case 2:
1094      lcd.print("MIN:"+String(round(min_ghi_si)));
1095      lcd.setCursor(12, 3);
1096      lcd.print("MAX:"+String(round(max_ghi_si)));
1097      break;
1098     case 3:
1099      lcd.print("MIN:"+String(round(min_dema_si)));
1100      lcd.setCursor(12, 3);
1101      lcd.print("MAX:"+String(round(max_dema_si)));
1102      break;
1103     case 4:
1104      break;
1105    }
1106    }else if(month<12 && hour<16){
1107      lcd.setCursor(1,3);
1108      if(month==0){
1109          aa=11;
1110        }else{
1111          aa=month-1;
1112        }
1113      switch (mode) {
1114        case 0:
1115          switch (hour) {
1116          case 0:
1117            lcd.print("Temp: "+String(round(avgtemp_4[aa])));
1118            break;
1119          case 1:
1120            lcd.print("Temp: "+String(round(avgtemp_5[aa])));
1121            break;
1122          case 2:
1123            lcd.print("Temp: "+String(round(avgtemp_6[aa])));
1124            break;
1125          case 3:
1126            lcd.print("Temp: "+String(round(avgtemp_7[aa])));
1127            break;
1128          case 4:
1129            lcd.print("Temp: "+String(round(avgtemp_8[aa])));
1130            break;
1131          case 5:
1132            lcd.print("Temp: "+String(round(avgtemp_9[aa])));
1133            break;
1134          case 6:
1135            lcd.print("Temp: "+String(round(avgtemp_10[aa])));
1136            break;
1137          case 7:
1138            lcd.print("Temp: "+String(round(avgtemp_11[aa])));
1139            break;
1140          case 8:
1141            lcd.print("Temp: "+String(round(avgtemp_12[aa])));
1142            break;
1143          case 9:
1144            lcd.print("Temp: "+String(round(avgtemp_13[aa])));
1145            break;
1146          case 10:
1147            lcd.print("Temp: "+String(round(avgtemp_14[aa])));
1148            break;
```

```
1149          case 11:
1150            lcd.print("Temp: "+String(round(avgtemp_15[aa])));
1151            break;
1152          case 12:
1153            lcd.print("Temp: "+String(round(avgtemp_16[aa])));
1154            break;
1155          case 13:
1156            lcd.print("Temp: "+String(round(avgtemp_17[aa])));
1157            break;
1158          case 14:
1159            lcd.print("Temp: "+String(round(avgtemp_18[aa])));
1160            break;
1161          case 15:
1162            lcd.print("Temp: "+String(round(avgtemp_19[aa])));
1163            break;
1164          }
1165          break;
1166        case 1:
1167          switch (hour) {
1168          case 0:
1169            lcd.print("Emis: "+String(round(avggrid_4[aa]))+unit_grid);
1170            break;
1171          case 1:
1172            lcd.print("Emis: "+String(round(avggrid_5[aa]))+unit_grid);
1173            break;
1174          case 2:
1175            lcd.print("Emis: "+String(round(avggrid_6[aa]))+unit_grid);
1176            break;
1177          case 3:
1178            lcd.print("Emis: "+String(round(avggrid_7[aa]))+unit_grid);
1179            break;
1180          case 4:
1181            lcd.print("Emis: "+String(round(avggrid_8[aa]))+unit_grid);
1182            break;
1183          case 5:
1184            lcd.print("Emis: "+String(round(avggrid_9[aa]))+unit_grid);
1185            break;
1186          case 6:
1187            lcd.print("Emis: "+String(round(avggrid_10[aa]))+unit_grid);
1188            break;
1189          case 7:
1190            lcd.print("Emis: "+String(round(avggrid_11[aa]))+unit_grid);
1191            break;
1192          case 8:
1193            lcd.print("Emis: "+String(round(avggrid_12[aa]))+unit_grid);
1194            break;
1195          case 9:
1196            lcd.print("Emis: "+String(round(avggrid_13[aa]))+unit_grid);
1197            break;
1198          case 10:
1199            lcd.print("Emis: "+String(round(avggrid_14[aa]))+unit_grid);
1200            break;
1201          case 11:
1202            lcd.print("Emis: "+String(round(avggrid_15[aa]))+unit_grid);
1203            break;
1204          case 12:
1205            lcd.print("Emis: "+String(round(avggrid_16[aa]))+unit_grid);
1206            break;
1207          case 13:
1208            lcd.print("Emis: "+String(round(avggrid_17[aa]))+unit_grid);
1209            break;
1210          case 14:
1211            lcd.print("Emis: "+String(round(avggrid_18[aa]))+unit_grid);
1212            break;
1213          case 15:
1214            lcd.print("Emis: "+String(round(avggrid_19[aa]))+unit_grid);
1215            break;
1216          }
1217          break;
1218        case 2:
1219          switch (hour) {
```

```
1220          case 0:
1221            lcd.print("Irrad: "+String(round(avgghi_4[aa]))+unit_ghi);
1222            break;
1223          case 1:
1224            lcd.print("Irrad: "+String(round(avgghi_5[aa]))+unit_ghi);
1225            break;
1226          case 2:
1227            lcd.print("Irrad: "+String(round(avgghi_6[aa]))+unit_ghi);
1228            break;
1229          case 3:
1230            lcd.print("Irrad: "+String(round(avgghi_7[aa]))+unit_ghi);
1231            break;
1232          case 4:
1233            lcd.print("Irrad: "+String(round(avgghi_8[aa]))+unit_ghi);
1234            break;
1235          case 5:
1236            lcd.print("Irrad: "+String(round(avgghi_9[aa]))+unit_ghi);
1237            break;
1238          case 6:
1239            lcd.print("Irrad: "+String(round(avgghi_10[aa]))+unit_ghi);
1240            break;
1241          case 7:
1242            lcd.print("Irrad: "+String(round(avgghi_11[aa]))+unit_ghi);
1243            break;
1244          case 8:
1245            lcd.print("Irrad: "+String(round(avgghi_12[aa]))+unit_ghi);
1246            break;
1247          case 9:
1248            lcd.print("Irrad: "+String(round(avgghi_13[aa]))+unit_ghi);
1249            break;
1250          case 10:
1251            lcd.print("Irrad: "+String(round(avgghi_14[aa]))+unit_ghi);
1252            break;
1253          case 11:
1254            lcd.print("Irrad: "+String(round(avgghi_15[aa]))+unit_ghi);
1255            break;
1256          case 12:
1257            lcd.print("Irrad: "+String(round(avgghi_16[aa]))+unit_ghi);
1258            break;
1259          case 13:
1260            lcd.print("Irrad: "+String(round(avgghi_17[aa]))+unit_ghi);
1261            break;
1262          case 14:
1263            lcd.print("Irrad: "+String(round(avgghi_18[aa]))+unit_ghi);
1264            break;
1265          case 15:
1266            lcd.print("Irrad: "+String(round(avgghi_19[aa]))+unit_ghi);
1267            break;
1268          }
1269          break;
1270
1271        case 3:
1272        switch (hour) {
1273          case 0:
1274            lcd.print("Demand: "+String(round(avgdema_4[aa]))+unit_dema);
1275            break;
1276          case 1:
1277            lcd.print("Demand: "+String(round(avgdema_5[aa]))+unit_dema);
1278            break;
1279          case 2:
1280            lcd.print("Demand: "+String(round(avgdema_6[aa]))+unit_dema);
1281            break;
1282          case 3:
1283            lcd.print("Demand: "+String(round(avgdema_7[aa]))+unit_dema);
1284            break;
1285          case 4:
1286            lcd.print("Demand: "+String(round(avgdema_8[aa]))+unit_dema);
1287            break;
1288          case 5:
1289            lcd.print("Demand: "+String(round(avgdema_9[aa]))+unit_dema);
1290            break;
```

```
1291          case 6:
1292            lcd.print("Demand: "+String(round(avgdema_10[aa]))+unit_dema);
1293            break;
1294          case 7:
1295            lcd.print("Demand: "+String(round(avgdema_11[aa]))+unit_dema);
1296            break;
1297          case 8:
1298            lcd.print("Demand: "+String(round(avgdema_12[aa]))+unit_dema);
1299            break;
1300          case 9:
1301            lcd.print("Demand: "+String(round(avgdema_13[aa]))+unit_dema);
1302            break;
1303          case 10:
1304            lcd.print("Demand: "+String(round(avgdema_14[aa]))+unit_dema);
1305            break;
1306          case 11:
1307            lcd.print("Demand: "+String(round(avgdema_15[aa]))+unit_dema);
1308            break;
1309          case 12:
1310            lcd.print("Demand: "+String(round(avgdema_16[aa]))+unit_dema);
1311            break;
1312          case 13:
1313            lcd.print("Demand: "+String(round(avgdema_17[aa]))+unit_dema);
1314            break;
1315          case 14:
1316            lcd.print("Demand: "+String(round(avgdema_18[aa]))+unit_dema);
1317            break;
1318          case 15:
1319            lcd.print("Demand: "+String(round(avgdema_19[aa]))+unit_dema);
1320            break;
1321          }
1322          break;
1323        case 4:
1324          break;
1325    }
1326    }else if(month<12 && hour==16){
1327      if(month==0){
1328          aa=11;
1329        }else{
1330          aa=month-1;
1331        }
1332      switch (mode) {
1333        case 0:
1334          temp_array[0]=avgtemp_4[aa];
1335          temp_array[1]=avgtemp_5[aa];
1336          temp_array[2]=avgtemp_6[aa];
1337          temp_array[3]=avgtemp_7[aa];
1338          temp_array[4]=avgtemp_8[aa];
1339          temp_array[5]=avgtemp_9[aa];
1340          temp_array[6]=avgtemp_10[aa];
1341          temp_array[7]=avgtemp_11[aa];
1342          temp_array[8]=avgtemp_12[aa];
1343          temp_array[9]=avgtemp_13[aa];
1344          temp_array[10]=avgtemp_14[aa];
1345          temp_array[11]=avgtemp_15[aa];
1346          temp_array[12]=avgtemp_16[aa];
1347          temp_array[13]=avgtemp_17[aa];
1348          temp_array[14]=avgtemp_18[aa];
1349          temp_array[15]=avgtemp_19[aa];
1350          break;
1351        case 1:
1352          temp_array[0]=avggrid_4[aa];
1353          temp_array[1]=avggrid_5[aa];
1354          temp_array[2]=avggrid_6[aa];
1355          temp_array[3]=avggrid_7[aa];
1356          temp_array[4]=avggrid_8[aa];
1357          temp_array[5]=avggrid_9[aa];
1358          temp_array[6]=avggrid_10[aa];
1359          temp_array[7]=avggrid_11[aa];
1360          temp_array[8]=avggrid_12[aa];
1361          temp_array[9]=avggrid_13[aa];
```

```
1362          temp_array[10]=avggrid_14[aa];
1363          temp_array[11]=avggrid_15[aa];
1364          temp_array[12]=avggrid_16[aa];
1365          temp_array[13]=avggrid_17[aa];
1366          temp_array[14]=avggrid_18[aa];
1367          temp_array[15]=avggrid_19[aa];
1368          break;
1369        case 2:
1370          temp_array[0]=avgghi_4[aa];
1371          temp_array[1]=avgghi_5[aa];
1372          temp_array[2]=avgghi_6[aa];
1373          temp_array[3]=avgghi_7[aa];
1374          temp_array[4]=avgghi_8[aa];
1375          temp_array[5]=avgghi_9[aa];
1376          temp_array[6]=avgghi_10[aa];
1377          temp_array[7]=avgghi_11[aa];
1378          temp_array[8]=avgghi_12[aa];
1379          temp_array[9]=avgghi_13[aa];
1380          temp_array[10]=avgghi_14[aa];
1381          temp_array[11]=avgghi_15[aa];
1382          temp_array[12]=avgghi_16[aa];
1383          temp_array[13]=avgghi_17[aa];
1384          temp_array[14]=avgghi_18[aa];
1385          temp_array[15]=avgghi_19[aa];
1386          break;
1387        case 3:
1388          temp_array[0]=avgdema_4[aa];
1389          temp_array[1]=avgdema_5[aa];
1390          temp_array[2]=avgdema_6[aa];
1391          temp_array[3]=avgdema_7[aa];
1392          temp_array[4]=avgdema_8[aa];
1393          temp_array[5]=avgdema_9[aa];
1394          temp_array[6]=avgdema_10[aa];
1395          temp_array[7]=avgdema_11[aa];
1396          temp_array[8]=avgdema_12[aa];
1397          temp_array[9]=avgdema_13[aa];
1398          temp_array[10]=avgdema_14[aa];
1399          temp_array[11]=avgdema_15[aa];
1400          temp_array[12]=avgdema_16[aa];
1401          temp_array[13]=avgdema_17[aa];
1402          temp_array[14]=avgdema_18[aa];
1403          temp_array[15]=avgdema_19[aa];
1404          break;
1405        case 4:
1406          break;
1407      }
1408      min_max(temp_array);
1409      lcd.print("MIN:"+String(round(minValue)));
1410      lcd.setCursor(12, 3);
1411      lcd.print("MAX:"+String(round(maxValue)));
1412
1413    }else if(month==12 && hour<16){
1414      lcd.setCursor(1,3);
1415        switch (mode) {
1416        case 0:
1417          switch (hour) {
1418          case 0:
1419            min_max(avgtemp_4);
1420            break;
1421          case 1:
1422            min_max(avgtemp_5);
1423            break;
1424          case 2:
1425            min_max(avgtemp_6);
1426            break;
1427          case 3:
1428            min_max(avgtemp_7);
1429            break;
1430          case 4:
1431            min_max(avgtemp_8);
1432            break;
```

```
1433          case 5:
1434            min_max(avgtemp_9);
1435            break;
1436          case 6:
1437            min_max(avgtemp_10);
1438            break;
1439          case 7:
1440            min_max(avgtemp_11);
1441            break;
1442          case 8:
1443            min_max(avgtemp_12);
1444            break;
1445          case 9:
1446            min_max(avgtemp_13);
1447            break;
1448          case 10:
1449            min_max(avgtemp_14);
1450            break;
1451          case 11:
1452            min_max(avgtemp_15);
1453            break;
1454          case 12:
1455            min_max(avgtemp_16);
1456            break;
1457          case 13:
1458            min_max(avgtemp_17);
1459            break;
1460          case 14:
1461            min_max(avgtemp_18);
1462            break;
1463          case 15:
1464            min_max(avgtemp_19);
1465            break;
1466          }
1467          lcd.print("MIN:"+String(round(minValue)));
1468          lcd.setCursor(12, 3);
1469          lcd.print("MAX:"+String(round(maxValue)));
1470          break;
1471        case 1:
1472          switch (hour) {
1473          case 0:
1474            min_max(avggrid_4);
1475            break;
1476          case 1:
1477            min_max(avggrid_5);
1478            break;
1479          case 2:
1480            min_max(avggrid_6);
1481            break;
1482          case 3:
1483            min_max(avggrid_7);
1484            break;
1485          case 4:
1486            min_max(avggrid_8);
1487            break;
1488          case 5:
1489            min_max(avggrid_9);
1490            break;
1491          case 6:
1492            min_max(avggrid_10);
1493            break;
1494          case 7:
1495            min_max(avggrid_11);
1496            break;
1497          case 8:
1498            min_max(avggrid_12);
1499            break;
1500          case 9:
1501            min_max(avggrid_13);
1502            break;
1503          case 10:
```

```
1504            min_max(avggrid_14);
1505            break;
1506          case 11:
1507            min_max(avggrid_15);
1508            break;
1509          case 12:
1510            min_max(avggrid_16);
1511            break;
1512          case 13:
1513            min_max(avggrid_17);
1514            break;
1515          case 14:
1516            min_max(avggrid_18);
1517            break;
1518          case 15:
1519            min_max(avggrid_19);
1520            break;
1521          }
1522          lcd.print("MIN:"+String(round(minValue)));
1523          lcd.setCursor(12, 3);
1524          lcd.print("MAX:"+String(round(maxValue)));
1525          break;
1526        case 2:
1527          switch (hour) {
1528          case 0:
1529            min_max(avgghi_4);
1530            break;
1531          case 1:
1532            min_max(avgghi_5);
1533            break;
1534          case 2:
1535            min_max(avgghi_6);
1536            break;
1537          case 3:
1538            min_max(avgghi_7);
1539            break;
1540          case 4:
1541            min_max(avgghi_8);
1542            break;
1543          case 5:
1544            min_max(avgghi_9);
1545            break;
1546          case 6:
1547            min_max(avgghi_10);
1548            break;
1549          case 7:
1550            min_max(avgghi_11);
1551            break;
1552          case 8:
1553            min_max(avgghi_12);
1554            break;
1555          case 9:
1556            min_max(avgghi_13);
1557            break;
1558          case 10:
1559            min_max(avgghi_14);
1560            break;
1561          case 11:
1562            min_max(avgghi_15);
1563            break;
1564          case 12:
1565            min_max(avgghi_16);
1566            break;
1567          case 13:
1568            min_max(avgghi_17);
1569            break;
1570          case 14:
1571            min_max(avgghi_18);
1572            break;
1573          case 15:
1574            min_max(avgghi_19);
```

```
1575            break;
1576          }
1577          lcd.print("MIN:"+String(round(minValue)));
1578          lcd.setCursor(12, 3);
1579          lcd.print("MAX:"+String(round(maxValue)));
1580          break;
1581        case 3:
1582        switch (hour) {
1583          case 0:
1584            min_max(avgdema_4);
1585            break;
1586          case 1:
1587            min_max(avgdema_5);
1588            break;
1589          case 2:
1590            min_max(avgdema_6);
1591            break;
1592          case 3:
1593            min_max(avgdema_7);
1594            break;
1595          case 4:
1596            min_max(avgdema_8);
1597            break;
1598          case 5:
1599            min_max(avgdema_9);
1600            break;
1601          case 6:
1602            min_max(avgdema_10);
1603            break;
1604          case 7:
1605            min_max(avgdema_11);
1606            break;
1607          case 8:
1608            min_max(avgdema_12);
1609            break;
1610          case 9:
1611            min_max(avgdema_13);
1612            break;
1613          case 10:
1614            min_max(avgdema_14);
1615            break;
1616          case 11:
1617            min_max(avgdema_15);
1618            break;
1619          case 12:
1620            min_max(avgdema_16);
1621            break;
1622          case 13:
1623            min_max(avgdema_17);
1624            break;
1625          case 14:
1626            min_max(avgdema_18);
1627            break;
1628          case 15:
1629            min_max(avgdema_19);
1630            break;
1631          }
1632          lcd.print("MIN:"+String(round(minValue)));
1633          lcd.setCursor(12, 3);
1634          lcd.print("MAX:"+String(round(maxValue)));
1635          break;
1636        case 4:
1637          break;
1638    }
1639
1640    }
1641  }
1642  void clearLCDLine(int line){
1643  for(int n = 0; n < 20; n++) {
1644  lcd.setCursor(n,line);
1645  lcd.print(" ");
```

```
1646 }
1647 }
1648
1649 void min_max(float ar[]){
1650   minValue = Min(ar[0], ar[1]);
1651   maxValue = Max(ar[0], ar[1]);
1652   for (int i = 2; i < 12; i++) {
1653     if(minValue>ar[a]){
1654       minValue=ar[a];
1655     }
1656     if (maxValue<ar[a]) {
1657       maxValue=ar[a];
1658     }
1659   }
1660 }
```