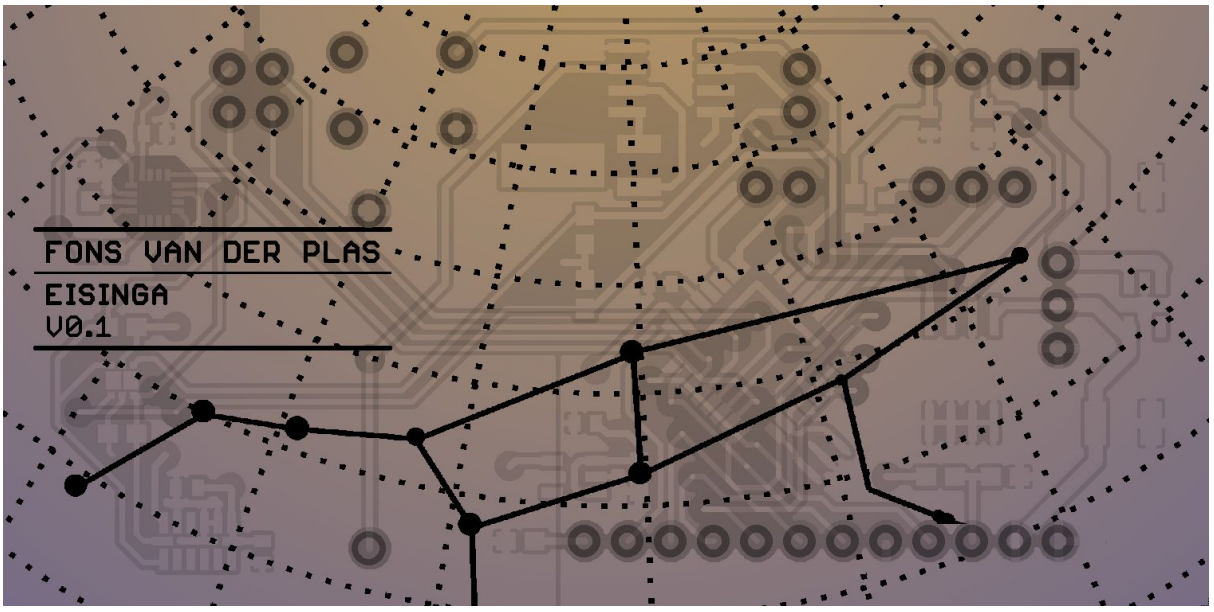


Eisinga

Designing an electronic constellation finder

Fons van der Plas

H. Mulders

March 2015

Preface

I would like to thank my project supervisor Hannie Mulders and my friends and family for supporting this project and their enthusiastic responses. Gijs van Dam and Ercan Kilic from Neways Electronics were kind enough to assemble and solder the entire prototype PCB. Finally, I would like to thank the online electronics and programming communities for their invaluable contribution.

Table of Contents

Introduction

The goal of this project was to create an electronic device capable of identifying stars, planets and constellations by pointing the device at them. While the orientation sensor readings proved to be too inaccurate to identify stars and planets, the device is capable of identifying constellations with reasonable accuracy. Currently, the functionality is software-bound, not hardware-bound. This means that the device can easily be improved in the future by adding new features and improving accuracy. While the development of this device has primarily been for educational reasons, it has some practical applications. For example, it might be useful tool for amateur astronomers trying to learn the constellations, or it might help when the visibility conditions are poor.
The project is called *Eisinga*, named after the dutch amateur astronomer Eise Eisinga (1744 - 1828).

This report is divided into two sections: astronomical research and development. The astronomical research was mainly focussed on the coordinate systems involved and astronomical databases. After writing this report, I started developing a prototype by using several commercial prototype boards and sensors. The goal of the first prototype was to be a starting point for writing software. When the software was completed, I started working on the PCB design. The second section of this report covers the final PCB design and some hardware and software elements.

The original project description (September 2014):

Overview

The goal of this project is to design and create a device that allows its user to identify the star or planet at which it is being pointed. It should be a low-cost device, to be used by users with intermediate experience. Example usage:

- Star gazing
- Astronomical photography
- Telescope orientation

Requirements

Device

Portable

The device must be able to run on batteries, without the need for an external power source. The device must also be able to operate without ever being connected to the internet.

Ease of use

The device should be designed for users with intermediate knowledge of our solar system and our night sky. The device should be intuitive, and usable without knowledge of its inner workings, and the amount of calibration should be minimal. The device must also be ambidextrous.
After identifying a star or planet, its information should be displayed to the user in a sensible way.

Low cost

The total cost of this device should be below €100,

Measurement accuracy

Measurements made by the device must be accurate until at least 2020, and the device must be usable at any location on the world.

Recognition

Stars

The device must be able to identify a finite number of bright, visible stars, along with its name and constellation.

Planets

All visible planets in our solar system should be identified.

Final Product

The final product will be the design of the device, a working prototype and a final report. This design must include:

- electronic schematics;
- PCB layout;
- all software used by the prototype

Astronomical research

Astronomy

The basic process of detecting a star is converting the device orientation to a coordinate in space. Although stars are stationary, the Earth rotates around its axis. Because we are standing on Earth, the sky seems to rotate. Given the observer’s position on Earth and the time and date, this rotation can be calculated.

Coordinate systems

There are two spherical coordinate systems primarily used for this application. All coordinate systems define the direction with two angles, relative to an origin, a fundamental plane and a primary direction.

Horizontal coordinate system

Origin: observer
Fundamental plane: horizon
Primary direction: north point on horizon

This coordinate system defines the orientation relative to the horizon (defined by the direction of gravitational pull) and the north pole.
A point is defined by its Azimuth (Az or A) and Altitude(Alt or a). Azimuth is the angle going eastward along the horizon, starting North. Altitude is the shortest angle between the point and the horizon with positive values above the horizon.

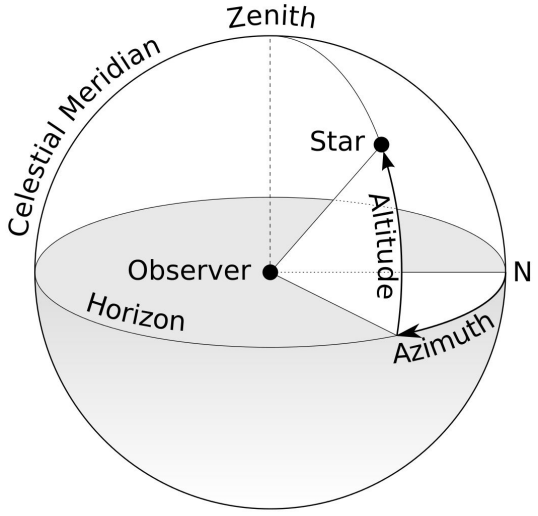


Figure 1: *The horizontal coordinate system. Source:* [1]

[1] 2011. File:Azimuth-Altitude schematic.svg - Wikimedia Commons. http://commons.wikimedia.org/wiki/File:Azimuth-Altitude_schematic.svg.

Equatorial coordinate system

Origin: center of Earth
Fundamental plane: plane perpendicular to Earth's rotational axis (equatorial plane)
Primary direction: vernal equinox (cross product of Earth's orbital axis and Earth's rotational axis)[2]

This coordinate system defines orientation independent of Earth's (daily) rotation.
A point is defined by its Right ascension (α) and Declination (δ). Right Ascension is the angle going eastward along the celestial equator. Declination is the shortest angle between the point and the celestial equator. Right ascension is often measured in units of time, not degrees.
Because this coordinate system is independent of Earth's rotation, stationary objects in the sky (such as stars) stay at fixed coordinates. For this reason, star positions are often measured in the equatorial coordinate system.

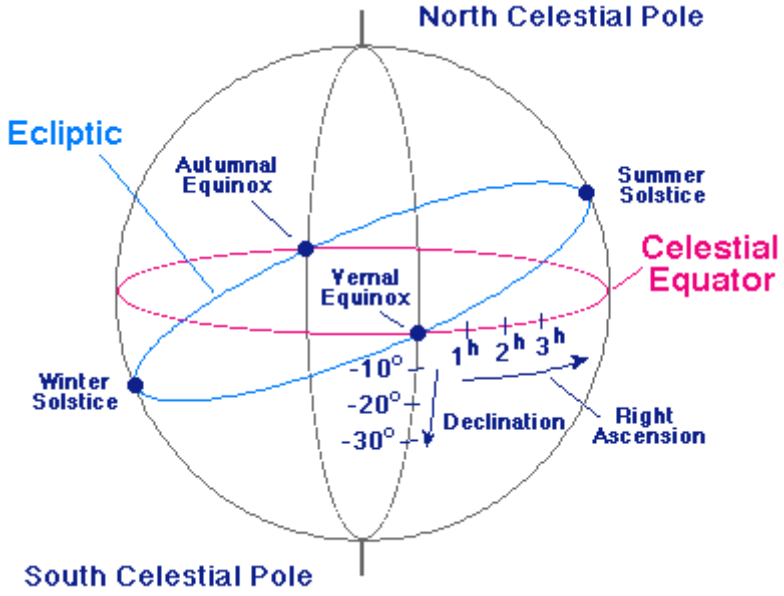


Figure 2: *The equatorial coordinate system. The path of the sun (ecliptic) is drawn in blue. Source:* [2]

Conversion

To convert between the horizontal and the equatorial coordinate system, the observer's latitude and local sidereal time (LST) are required. The local sidereal time measures Earth's rotation relative to the stars, not the Sun. Because of Earth's orbit around the sun, a sidereal day is slightly shorter than a solar day, by a difference of about 3 minutes and 56 seconds.[3]
Local sidereal time is defined as the right ascension of a celestial body at the local meridian (with an Azimuth of 0°). Local sidereal time can be calculated using the observer's longitude and the Greenwich Sidereal Time (GST):

```
GST = 18.697374558 + 24.06570982441908 * D
LST = GST + λ / 15°
```

[2] Celestial Coordinate System. http://csep10.phys.utk.edu/astr161/lect/time/coordinates.html.
[3] Timekeeping. http://csep10.phys.utk.edu/astr161/lect/time/timekeeping.html.

Where LMST and GMST are expressed in hours, λ is the observer’s longitude in degrees and D is the number of UT days since January 1, 2000 at noon, as a fraction.[4,5]

Equatorial → horizontal

To convert equatorial coordinates to horizontal coordinates, the following algorithm is used, developed by Paul Schlyter in 1979. This algorithm was designed with optimal accuracy and minimal computational costs.[6]

```
HA = RA – LST

xhor = cos(HA) * cos(Decl) * sin(lat) – sin(Decl) * cos(lat)
yhor = sin(HA) * cos(Decl)
zhor = cos(HA) * cos(Decl) * cos(lat) + sin(Decl) * sin(lat)

az  = atan2(yhor, xhor) + 180_degrees                          // Azimuth
alt = asin(zhor) = atan2(zhor, sqrt(xhor * xhor + yhor * yhor)) // Altitude
```

Where HA is the hour angle, between -12 and 12 hours (converted to radians).

Horizontal → equatorial

The following conversion from horizontal to equatorial coordinates is derived from the Astronomical Triangle.[7]

```
Decl = asin(sin(alt) * sin(lat) + cos(alt) * cos(lat) * cos(az))
HA = atan2(sin(az), sin(alt) * cos(lat) / cos(alt) – cos(yaw) * sin(lat))

RA = LST + HA
```

Refraction

Because the atmosphere has a higher density than the vacuum of space, light from stars gets refracted, which decreases the apparent altitude. The amount of refraction is zero at the zenith, and 0.30° at 2° altitude.
Atmospheric refraction can be approximated using the atmospheric pressure, temperature and altitude, using the following formula:

$$Re = Pr * (0.1594 + 0.0196 * a + 0.00002 * a^2)/(T * (1 + 0.505 * a + 0.0845 * a^2))$$

Where Re is the refraction towards the horizon in degrees, Pr is the atmospheric pressure in millibars, T is the temperature in Kelvin and a is the altitude in degrees.[8]

---

[4] 2005. Astronomical Times. http://www.cv.nrao.edu/~rfisher/Ephemerides/times.html.
[5] 2007. Approximate Sidereal Time - U.S. Naval Observatory ... http://aa.usno.navy.mil/faq/docs/GAST.php.
[6] 2003. Computing planetary positions - Paul Schlyters hemsida. http://www.stjarnhimlen.se/comp/ppcomp.html.
[7] Conversion between horizontal and eq. http://star-www.st-and.ac.uk/~fv/webnotes/chapter7.htm.
[8] Refraction Calculator. http://wise-obs.tau.ac.il/~eran/Wise/Util/Refraction.html.

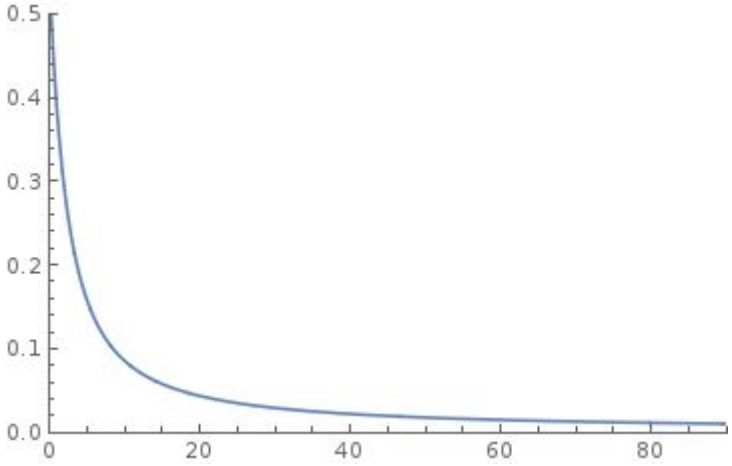


Figure 3: *Refraction (y) and altitude (x), both in degrees at regular conditions. Source: own work.*

Star positions

While number of recorded stars is enormous and expanding daily, this device only needs stars visible to the human eye. To gather this data, a list of the brightest stars is used. Specifically, the Yale Bright Star catalog is used, which contains data for every star with a stellar (apparent) magnitude of 6.5 or lower (stars with a stellar magnitude above 6.5 are invisible to the human eye).
This is a list of 9110 stars, taking up 285 kB of computer memory.[9] It might be required to use a smaller list, depending on the software limits.

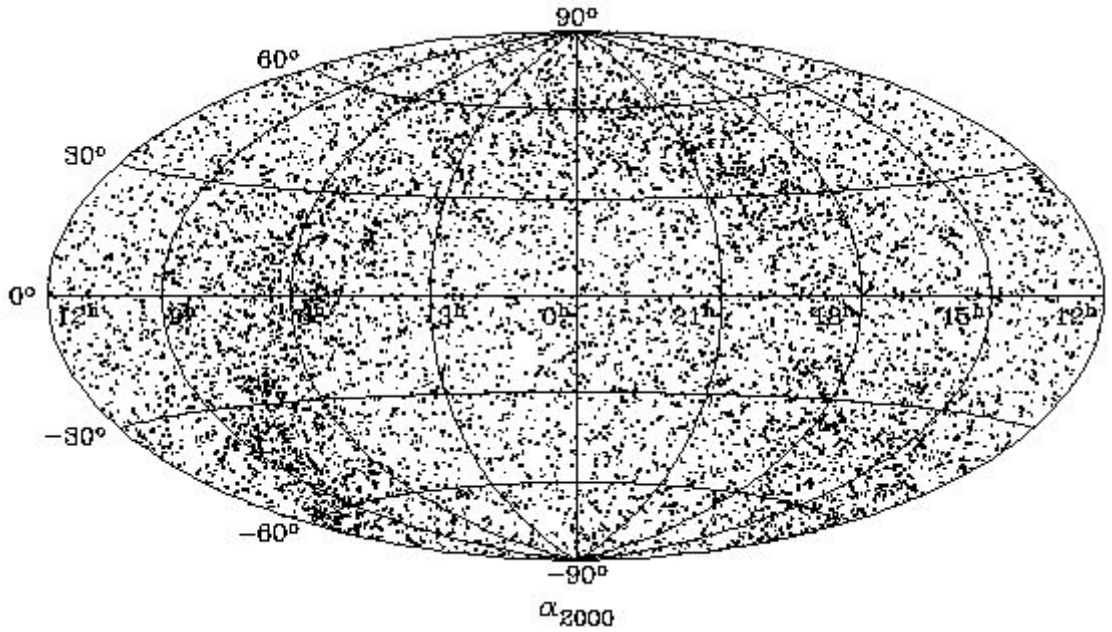


Figure 4: *A map of all stars in the Yale Bright Star catalog. Source:* [9]

Planetary positions

To compute planetary positions, the algorithm developed by Paul Schlyter is used. He has written an extensive report and example code in C, which will be used in the software, with minor changes to optimise the algorithm for this application.

[9] 2005. Yale Bright Star Catalog 5 (BSC5). http://tdc-www.harvard.edu/catalogs/bsc5.html.

Algoritm report: http://www.stjarnhimlen.se/comp/ppcomp.html
Sample code: http://www.stjarnhimlen.se/comp/tutorial.html

Constellation coordinates

In 1922, the boundaries between the 88 constellations were devised by the International Astronomy Union along vertical and horizontal lines of right ascension and declination.[10,11] The device uses an algorithm which splits the constellations into rectangles with no upper declination boundary. Therefore, every rectangle is defined by lower and upper right ascension boundaries, and a lower declination boundary. All rectangles are sorted in order of their lower declination boundaries, going from 90° to -90°. This database is stored in program memory, and is used to determine the constellation of a given coordinate as follows: the algorithm iterates through all rectangles in the predefined order (as described previously). Every rectangle is compared to the given coordinate, and the first rectangle to intersect with the coordinate corresponds with the output constellation.
Because no (computationally) expensive operations are performed, the algorithm is very quick. Furthermore, this algorithm performs best on constellations close to the celestial North (Polaris), which are more commonly seen in the Northern hemisphere.

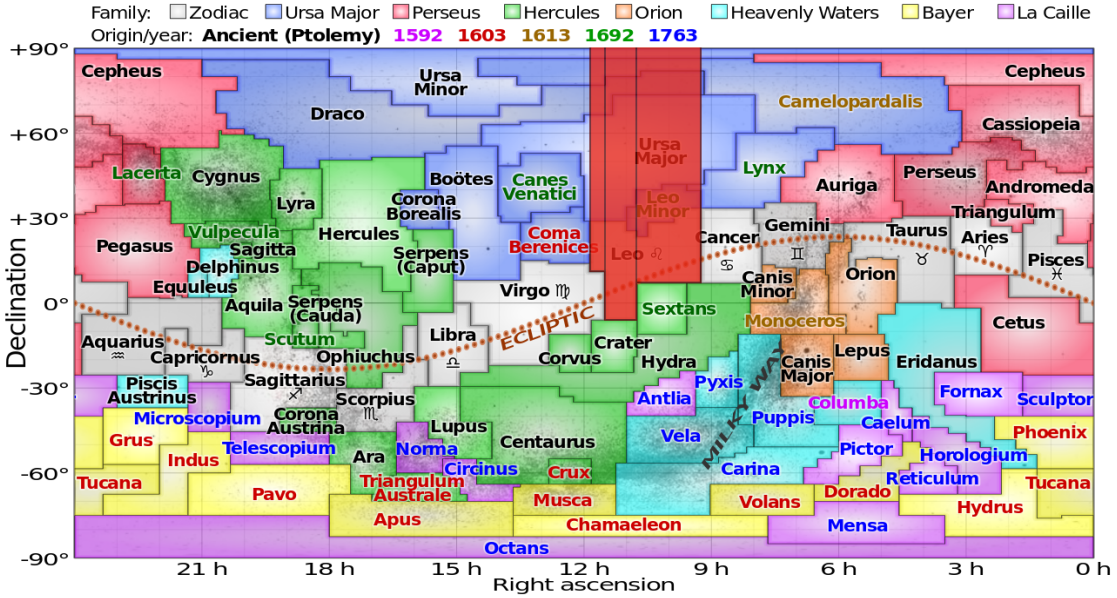


Figure 5: *An equirectangular map of the 88 constellations, as devised by the IAU. The three rectangles used for Leo by the constellation algorithm are colored red. Source: [12], modified.*

[10] 2010. Origin of the modern constellation boundaries - Ian Ridpath. http://www.ianridpath.com/boundaries.htm.
[11] 2013. The Constellations | IAU. http://www.iau.org/public/themes/constellations/.
[12] 2013. File:Constellations ecliptic equirectangular plot.svg ... http://commons.wikimedia.org/wiki/File:Constellations_ecliptic_equirectangular_plot.svg.

Development

Hardware research

The device can be abstracted using the following flowchart:

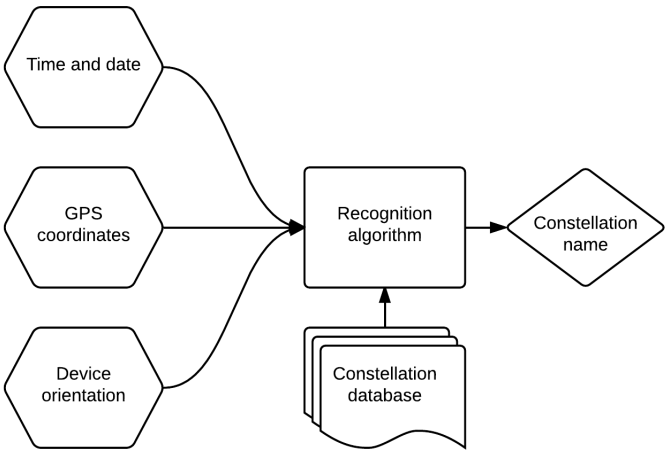


Figure 6: *Flowchart*

Each element in figure 6 requires a specific hardware component, with the exception of “GPS coordinates”, as the GPS coordinates are set by the user.

Flowchart element	**Hardware component**
Time and date	Real time clock (RTC)
Device orientation	Orientation sensors
Recognition algorithm	Microprocessor
Constellation database	Storage element (EEPROM)
Constellation name	Display (LCD)
Other	Power management, circuit board (PCB)

Real time clock (RTC)

To keep the current time, a combination of an electronic oscillator and a clock module is used. Because the device will not be powered at all times, a separate battery is required to power the clock module. The combination of an electronic oscillator, a clock module (IC) and a battery is called a real time clock (RTC).

Orientation sensors

To determine device rotation relative to Earth the device needs to use a combination of orientation sensors. The following sensors are available:

Accelerometer

An accelerometer senses acceleration relative to freefall in 3 dimensions. When an accelerometer is laying flat on Earth, the z-axis will read 1g, and the x and y axis will read negligible values. When the accelerometer is rotated, the magnitude of the vector will remain at 1g.
The output for each axis equals the cosine of the angle between that axis and the directional vector of gravity.
An accelerometer is used to sense orientation relative to Earth’s gravitational field, but not relative to North. Only two axis of freedom can be derived from accelerometer data.

Magnetometer

A magnetometer senses the direction and magnitude of the magnetic field at the point of the sensor, and outputs a 3D vector. For each axis, the output equals the cosine of the angle between the axis and the magnetic field, multiplied by the magnitude of the magnetic field.
A magnetometer is used to sense orientation relative to Earth’s geomagnetic field. While only two axis of freedom can be derived from magnetometer data, this sensor can be combined with an accelerometer to sense full orientation.

Gyroscope

A gyroscope measures rotational velocity in 3 axis. Gyroscope measurements are unnecessary in this device, because rotational velocity during a measurement should be negligible.

Optical

Another option is to use a camera to recognise a star. However, a camera would be too unreliable and expensive. Furthermore, it takes at least 10 seconds of exposure for a high-end camera to take a proper image of the sky, which will require the device to be stationary.

Mechanical

Using a two rotary encoders, the azimuth and altitude can be determined directly. This requires the device to be standing on a tripod facing North. While this approach is quite accurate, the device becomes less portable, and requires the user to position the tripod very precisely.

Microprocessor

Because the device functionality is quite complex, a microprocessor is needed to interpret sensor data and display information to the user. There is a wide variety of microcontrollers available for different purposes. One notable example is the 8-bit Atmel ATmega AVR series, as it is used in the open source Arduino project. Arduino is a simple development board with an ATmega processor, which is flashed with the Arduino bootloader. This allows the microcontroller to communicate and be programmed over serial communication (USB), using regular C++ code. The ATmega chip is capable of I^2C and SPI communication, which allows it to connect to other devices such as orientation sensors or LCDs.
Because the Arduino project is open source, the bootloader and schematics are freely available.

Storage element

A storage element is required to store the astronomical data required for recognition, as the microprocessor is unlike to have enough program memory. The storage element needs to be able to store at least 25 kB of data to fit the constellation database (3,6 kB) and the star database (around 20 kB). Furthermore, the storage element needs to be fast, as database is not stored in RAM during runtime. This means that it might be required to load the entire database into the microprocessor to perform a single recognition.

Software

Orientation sensor interpretation

For every measurement, the device orientation is derived from the accelerometer and magnetometer readings. The accelerometer and magnetometer are both 3D sensor, and measure the force or field strength relative to three axes: *X*, *Y* and *Z*. Because the strength of gravity (and thus acceleration) and the strength of the geomagnetic field are constant, the calibrated readings of these sensors lay on the surface of a sphere, centered at the origin. In the case of an accelerometer, the radius of this sphere equals *g*, the gravitational acceleration. In the case of a magnetometer, the radius equals *B*, the geomagnetic field strength at the location of measuring.
When the device is laying flat on a surface, facing North, the sensor readings are defined:

$$\boldsymbol{G}_r = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} \qquad \textbf{Equation 1}$$

$$\boldsymbol{B}_r = B \begin{pmatrix} \cos\delta \\ 0 \\ \sin\delta \end{pmatrix} \qquad \textbf{Equation 2}$$

where δ is the angle of geomagnetic inclination, which is the angle between the horizon and the direction of the geomagnetic field (its altitude).
During measurement, the device will have an arbitrary rotation, defined by three factors: yaw (azimuth), pitch (altitude) and roll. Because the direction of gravity and the direction of the geomagnetic field are different (not parallel), the device rotation can be derived from the accelerometer and magnetometer readings.
First, roll (ϕ) and pitch (θ) angles are derived from the accelerometer reading. Because yaw is a rotation around the same axis as the direction of gravity, yaw can not be derived from accelerometer readings.

$$\tan(\phi) = \left(\frac{G_{py}}{G_{pz}}\right) \qquad \textbf{Equation 3}$$

$$\tan(\theta) = \left(\frac{-G_{px}}{G_{py}\sin\phi + G_{pz}\cos\phi}\right) \qquad \textbf{Equation 4}$$

The pitch and roll values are then used to derotate the magnetometer reading, which results in a point laying on a circle parallel to the *X,Y*-plane, as the *Z* value of this reading is constant. The position on the sphere, as defined by the *X* and *Y* values, is used to derive the yaw angle (Ψ).

$$\tan(\psi) = \left(\frac{(B_{pz} - V_z)\sin\phi - (B_{py} - V_y)\cos\phi}{(B_{px} - V_x)\cos\theta + (B_{py} - V_y)\sin\theta\sin\phi + (B_{pz} - V_z)\sin\theta\cos\phi}\right) \qquad \textbf{Equation 5}$$

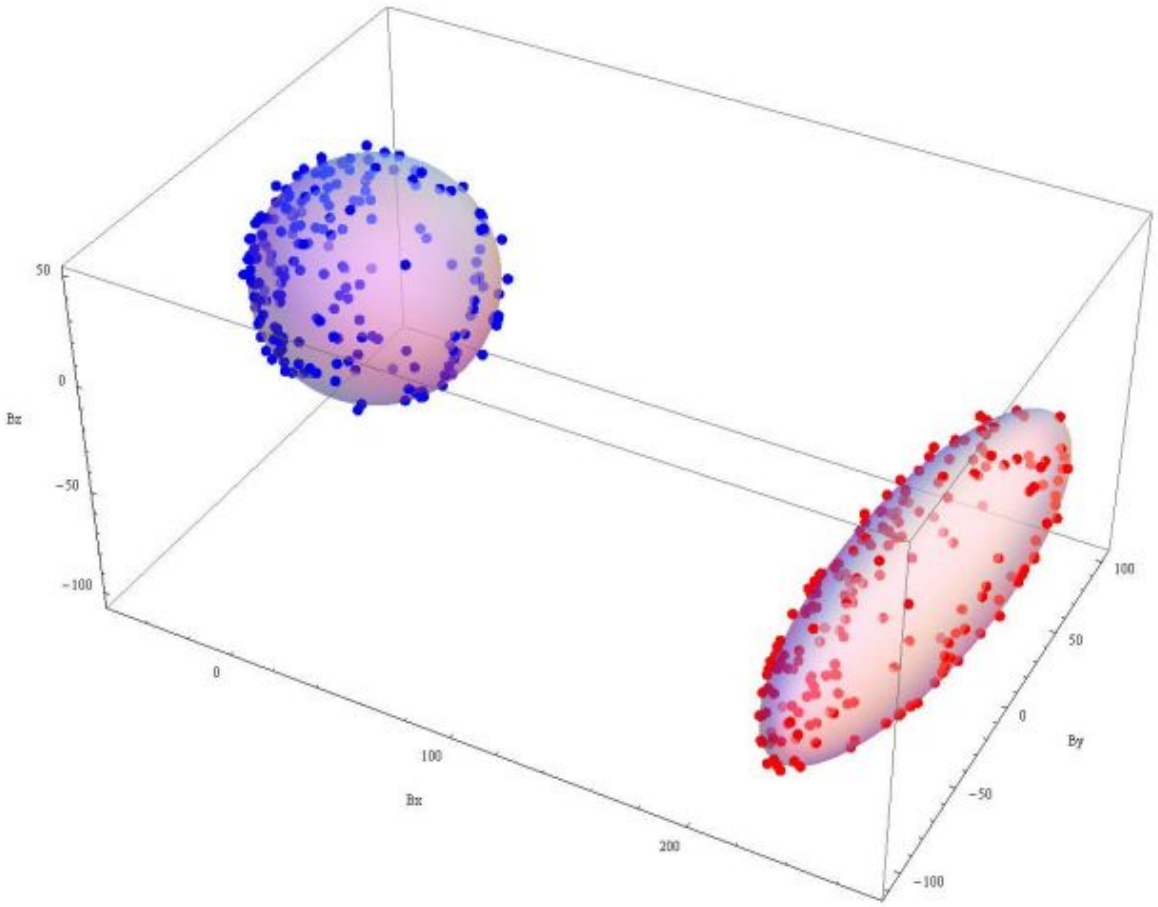
Source: [13]

[13] Ozyagcilar, Talat. 2012. Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. *Freescale semiconductor, AN* 4248.

Orientation sensor calibration

The previous section describes how the yaw, pitch and roll angles are derived from the accelerometer and magnetometer readings. This algorithm requires these readings to lay on the surface of a sphere centered at the origin. However, raw sensor readings rarely conform to this requirement. Because of various factors, the readings lay on the surface of an ellipsoid, which is not centered at the origin. In the case of an accelerometer, the sensor is calibrated after production, but over time the sensor properties change slightly, which makes the original calibration inaccurate. The magnetometer faces the same problem and in addition, the magnetometer is influenced by electric fields, which are generated by the surrounding electronics and the chip itself. Because these sources of interference move and rotate in exactly the same way as the magnetometer itself, this interference only causes an offset, and not a rotation of sensor readings (which would be impossible to calibrate). This offset is called the hard-iron offset.

In addition the the hard-iron offset, the sensor readings are also affected by the soft-iron effects, which cause the measurements to lay on the surface of an ellipsoid, rather than the surface of a sphere. The soft-iron effects have three primary sources: different sensitivities among the *X, Y* and *Z* axes, lack of perfect orthogonality between the *X, Y* and *Z* axes, and the effects of normally non magnetized ferromagnetic components on the PCB. These effects can be approximated by a 3 by 3 matrix, *W*, that transforms the spherical values into ellipsoidal ones.



Firgure 7: *A scatter plot of raw measurements (red) laying on the surface of an ellipsoid, and their calibrated equavalents (blue) laying on the surface of a sphere centered at the origin. Source:* [14]

(The following algorithm describes the magnetometer calibration, which is identical to the accelerometer calibration.) To compensate for these problems, four components need to be determined by means of calibration: the 3-component measurement offset *V* and the geomagnetic field strength, *B*. The measured sensor value, B_p , influenced by the hard-iron offset, *V*, the soft-iron matrix, *W*, and the arbitrary device rotation (*R* is a rotation matrix):

$$\boldsymbol{B}_p = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)B\begin{pmatrix}\cos\delta\\0\\\sin\delta\end{pmatrix} + \boldsymbol{V} \qquad \textbf{Equation 6}$$

Because the calibrated values lay on the surface of a sphere, with a radius equal to the geomagnetic field strength, *B*:

$$\left\{\boldsymbol{W}^{-1}(\boldsymbol{B}_p - \boldsymbol{V})\right\}^T \boldsymbol{W}^{-1}(\boldsymbol{B}_p - \boldsymbol{V}) = B^2 \qquad \textbf{Equation 7}$$

In the process of finding the hard-iron offset, *V*, the soft-iron effects can be ignored. Equation 7 then simplifies to:

$$(\boldsymbol{B}_p - \boldsymbol{V})^T(\boldsymbol{B}_p - \boldsymbol{V}) = B^2 \qquad \textbf{Equation 8}$$

$$\Rightarrow \boldsymbol{B}_p{}^T\boldsymbol{B}_p - 2\boldsymbol{B}_p{}^T\boldsymbol{V} + \boldsymbol{V}^T\boldsymbol{V} - B^2 = 0 \qquad \textbf{Equation 9}$$

Because the raw measurements are determined on an experimental basis, the left side of equation 9 does not equal 0 exactly, but has a residual error. The residual error, *r[i]*, of the *i*-th measurement
normalized then equals:

$$r[i] = \left(B_{px}[i]^2 + B_{py}[i]^2 + B_{pz}[i]^2\right) - \begin{pmatrix}B_{px}[i]\\B_{py}[i]\\B_{pz}[i]\\1\end{pmatrix}^T\begin{pmatrix}2V_x\\2V_y\\2V_z\\B^2 - V_x^2 - V_y^2 - V_z^2\end{pmatrix} \qquad \textbf{Equation 10}$$

[14] Ozyagcilar, Talat. "Calibrating an ecompass in the presence of hard and soft-iron interference." *Freescale Semiconductor Ltd* (2012).

To find the best fit for *V* and *B*, the total residual error is minimized. The total residual error is defined as the sum of the squares of each measurement’s residual error:

$$P = r[0]^2 + r[1]^2 + \ldots + r[M-1]^2 \qquad \textbf{Equation 11}$$

where *M* is the number of measurements. The solution to this problem is defined as *β*, which contains all four calibration components, and is equal to a function of *X* (*M* by 4) and Y (*M* by 1):

$$\beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{pmatrix} = \begin{pmatrix} 2V_x \\ 2V_y \\ 2V_z \\ B^2 - V_x^2 - V_y^2 - V_z^2 \end{pmatrix} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \qquad \textbf{Equation 12}$$

where:

$$\boldsymbol{X}^T\boldsymbol{X} = \sum_{i=0}^{M-1} \begin{pmatrix} B_{px}[i]^2 & B_{px}[i]B_{py}[i] & B_{px}[i]B_{pz}[i] & B_{px}[i] \\ B_{px}[i]B_{py}[i] & B_{py}[i]^2 & B_{py}[i]B_{pz}[i] & B_{py}[i] \\ B_{px}[i]B_{pz}[i] & B_{py}[i]B_{pz}[i] & B_{pz}[i]^2 & B_{pz}[i] \\ B_{px}[i] & B_{py}[i] & B_{pz}[i] & 1 \end{pmatrix} \qquad \textbf{Equation 13}$$

$$\boldsymbol{X}^T\boldsymbol{Y} = \sum_{i=0}^{M-1} \begin{pmatrix} B_{px}[i](B_{px}[i]^2 + B_{py}[i]^2 + B_{pz}[i]^2) \\ B_{py}[i](B_{px}[i]^2 + B_{py}[i]^2 + B_{pz}[i]^2) \\ B_{pz}[i](B_{px}[i]^2 + B_{py}[i]^2 + B_{pz}[i]^2) \\ B_{px}[i]^2 + B_{py}[i]^2 + B_{pz}[i]^2 \end{pmatrix} \qquad \textbf{Equation 14}$$

From this, *β* can be calculated, which contains the four calibration components. Because equations 13 and 14 equal to the sum of M individual matrices, each defined by the *i*-th measurement, the calibration process can have an undetermined number of measurements, without a storage overhead.
Using *β* every future measurement, B_p, can be calibrated by:

1. subtracting *V* from B_p to compensate for hard-iron effects, resulting in an ellipsoid centered at the origin, and then
2. normalizing the resulting vector to unit length, to compensate for the soft-iron effects, resulting in a sphere centered at the origin.

Source: [14]

Final design

The final design (v0.2) consists of three parts: a 9V battery, a commercial 16 by 2 character LCD and a PCB. The PCB was designed specifically for this purpose, and contains all required components. It holds the microcontroller, 2 orientation sensors, an RTC module, an EEPROM module, a regulated 5V power circuit, a regulated 3.3V power circuit and 30 unconnected pins (see PCB section). The accelerometer and magnetometer were both chosen because they were available on a cheap IMU (inertial measurement unit) prototyping board called GY-80, which was originally intended for drone design. Both sensors communicate over the I^2C protocol, which is a common protocol used by microcontrollers and digital sensors. However, the accelerometer and the magnetometer operate on 3.3V, while most other devices operate op 5V. Because of this, there is a second power circuit on the board that converts the regulated 5V to 3.3V. To allow communication between the 5V microcontroller and the 3.3V sensors, a logic level shifter was required. This is a combination of an N-channel MOSFET transistor and two 4.7 kΩ pull-up resistors (bottom left of figure 8). The RTC and EEPROM module were both chosen because they communicate over I^2C and because of their low price and high availability.

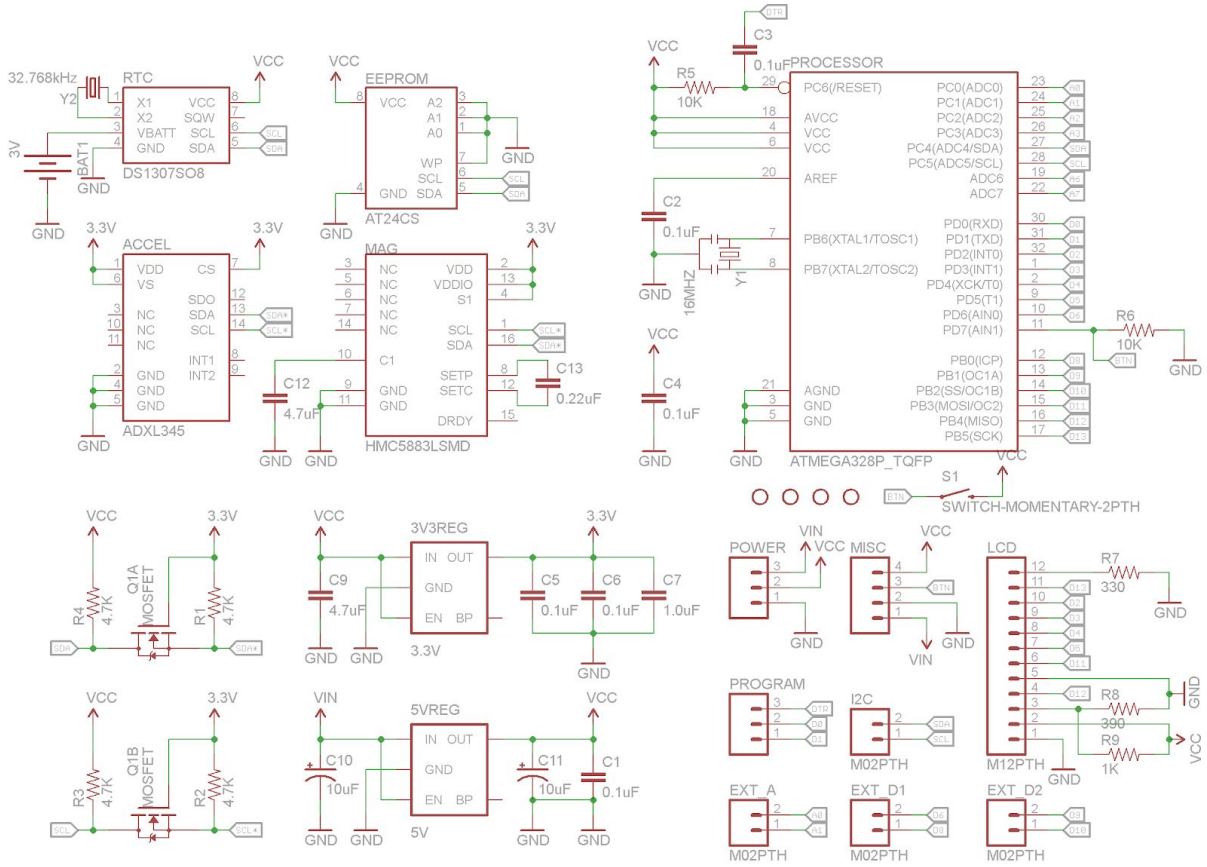


Figure 8: *The final schematics. Source: own work.*

PCB

The final PCB design is a 2-layer PCB which measures at 3.5 x 7.1 mm. All components are on one side, and most components are surface-mounted. There are 30 unconnected pins: 12 pins connect the LCD to the board and 2 pins connect the battery to the board. There are 16 remaining pins: 2 pins connect to an external button, 3 pins connect to a second external power source or another 5V device, 3 pins connect to an AVR programmer (to install the Arduino bootloader), 2 I^2C pins allow the included sensors to be used by another device, or to expand the board by adding more sensors, and 2 analog and 4 digital pins allow the device to be expanded even further.
Before the assembled device is usable, the ATmega328 needs to be flashed with the Arduino bootloader, and programmed with the Eisinga program. To load a bootloader onto the chip, the pins GND, VCC, D11, D12, D13 and DTR. The current LCD pins were intentionally chosen to match all of these pins, except DTR, which is found on the board elsewhere. After the bootloader is flashed onto the microprocessor, it can be programmed over serial, which uses pins GND, VCC, D0, D1 and DTR. Because the pins D0 and D1 cannot be used to drive the LCD, these pins are also added to the board.

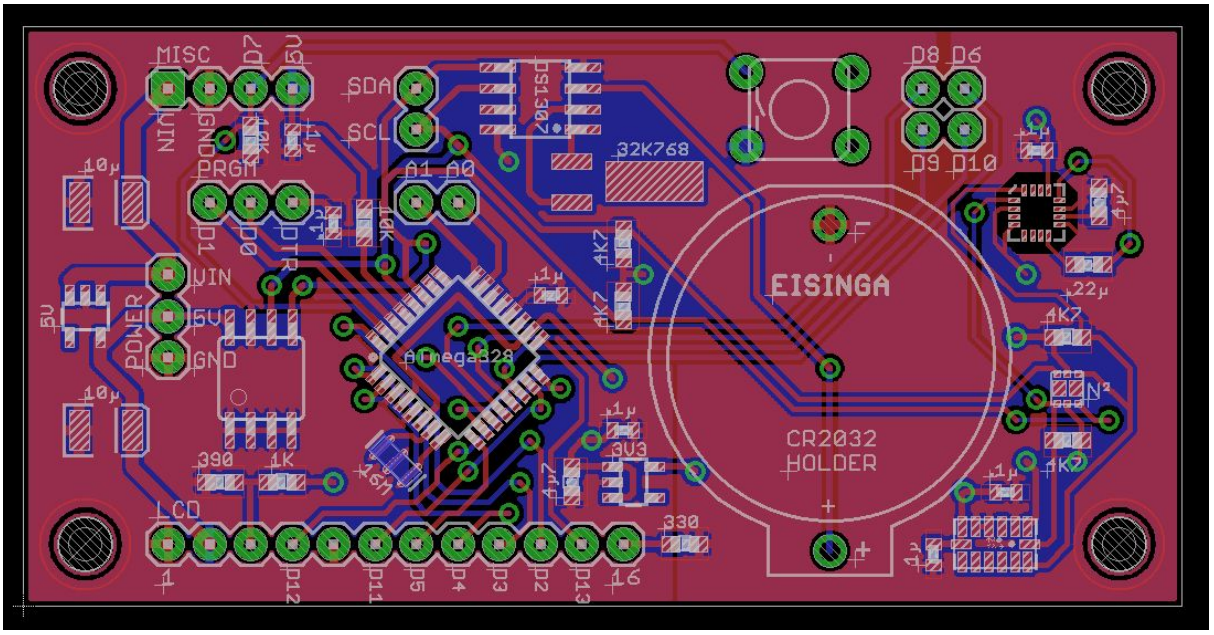


Figure 9: *The final PCB design. Top copper is colored red, bottom copper is colored blue. Vias and pads are colored green. The silkscreen is colored white. Source: own work.*

PCB design

While designing a PCB, a lot of factors need to be taken into consideration. After creating the schematics, the first step is to approximately place the components on the board. When designing the PCB for this device, the following factors were taken into account:

Component grouping

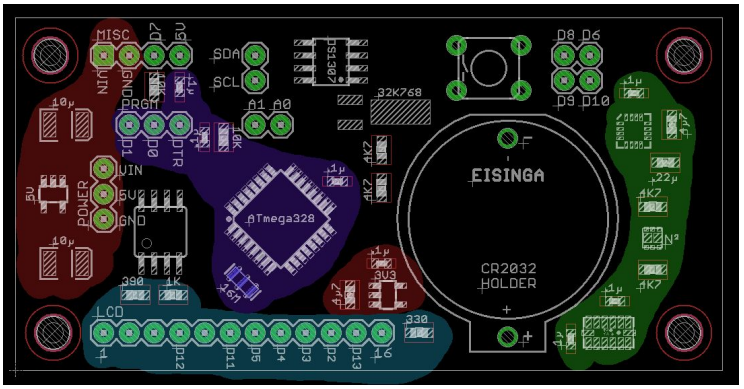


Figure 10: *Related components are grouped together, to minimize trace lengths. Colored groups: power management (red), microprocessor (purple), LCD connector (blue) and 3.3V sensors (green). The 3.3V regulator (designated as '3V3') is in the center. Source: own work.*

5V and 3.3V

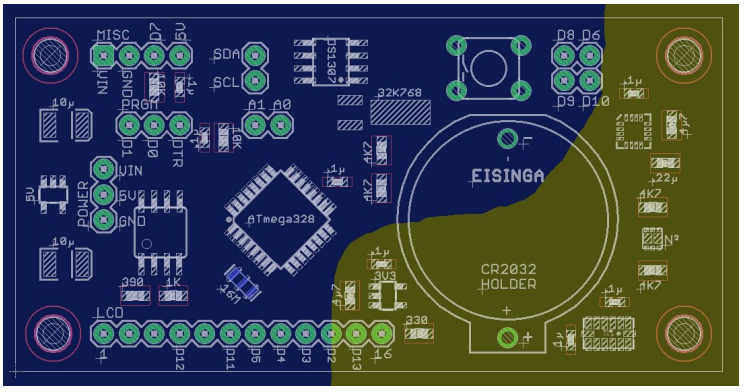


Figure 11: *Most components require a 5V power source, but the accelerometer and magnetometer require a 3.3V power source. Because of this difference, the board is divided into two sections: a 5V section (blue) and a 3.3V section (yellow). Source: own work.*

Magnetometer

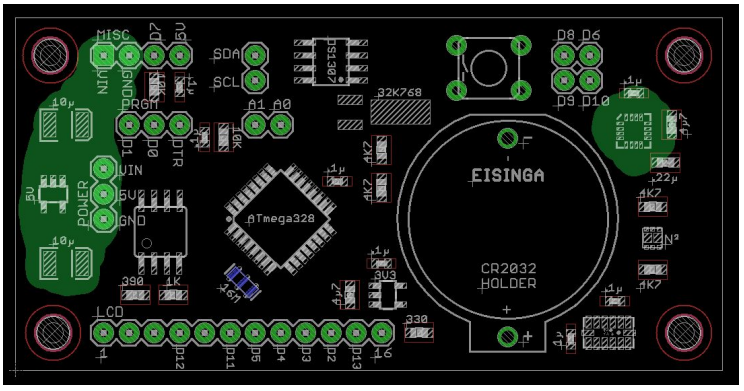


Figure 12: *The magnetometer readings are influenced by the (geo)magnetic field as well as the electric field. To reduce electric field noise, and to thus improve magnetometer performance, the magnetometer (right) and the power management circuit (left) are placed on opposite sides of the board. The RTC battery was intentionally placed between the magnetometer and the rest of the board, as it might absorb some of the magnetic noise. The current placement of the three magnetometer capacitors (0.1 µF, 4.7 µF and 0.22 µF) was recommended by the associated datasheet. Source: own work.*

Routing

Traces between the connecting leads connect to each other, in accordance with the schematics. The exact placement of these traces, a process called 'routing', is a difficult problem. It is often possible to route the PCB automatically using an automated routing algorithm, but manually routing the PCB often gives a better result.[15] Besides, manually routing the PCB has been a very interesting and informative experience.

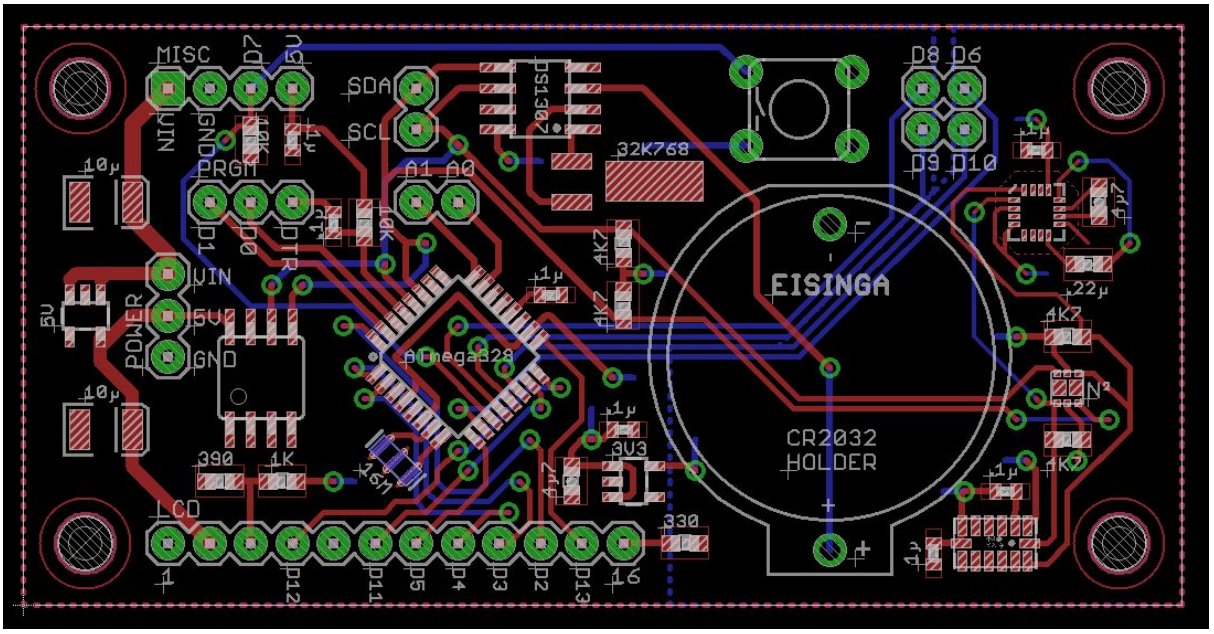


Figure 13: *All signals are routed, except the ground, 5V and 3.3V lines. Source: own work.*

The ground, 5V and 3.3V lines were routed after all other lines, to save time. After all other lines were routed, the top layer was filled with a ground pour (a connecting layer of copper which fills all available empty space), and the bottom layer was filled with two separate pours: the left and right side of the board were filled with 5V and 3.3V pours, respectively. The pours are visible in figure 9.

Silk screen

Most PCB manufacturers allow a silk layer on top of the PCB, which is a white ink layer meant to help in the soldering process and to improve repairability. It is often used to designate component positions and values, to display information about the board (like the manufacturing date and board version) and to decorate the board.
The top silk layer on this board contains information related to components, such as resistor and capacitor values, and component orientation (polarity). Because all components are on one side of the board, a bottom silk layer is not necessary. However, because most manufacturers allow a bottom silk layer with no extra costs, I designed a decorative bottom layer, which should give some idea of the purpose and focus area of this device.

[15] 2013. Using EAGLE: Board Layout - learn.sparkfun.com. https://learn.sparkfun.com/tutorials/using-eagle-board-layout/routing-the-board.

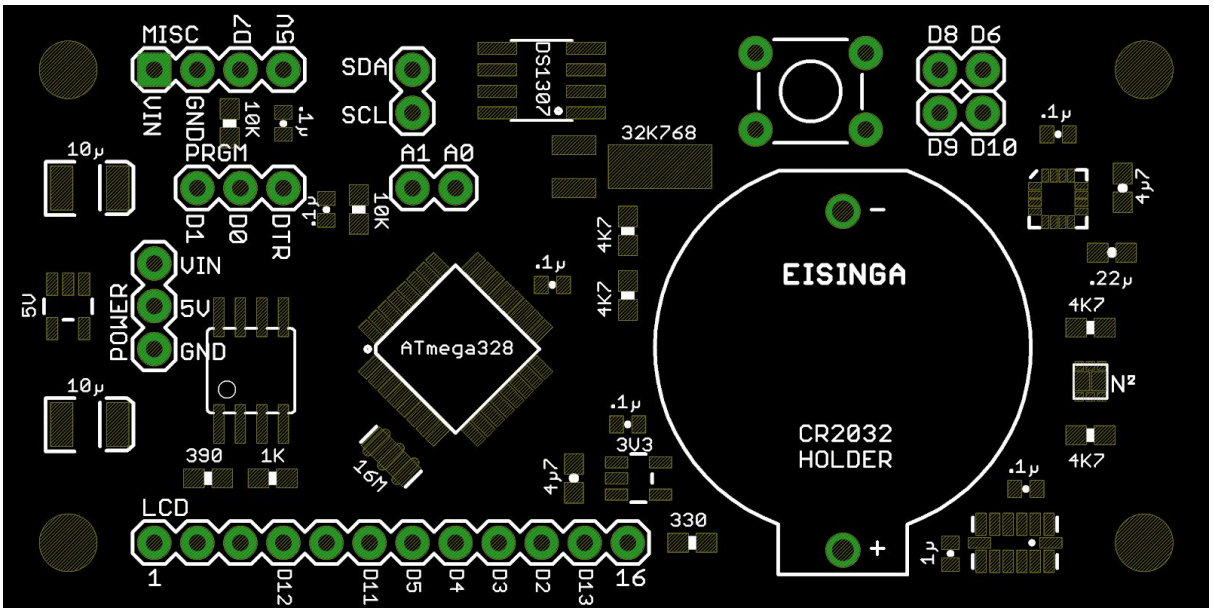


Figure 14: *The top silk layer. Solder pads are also displayed in orange and green. Source: own work.*

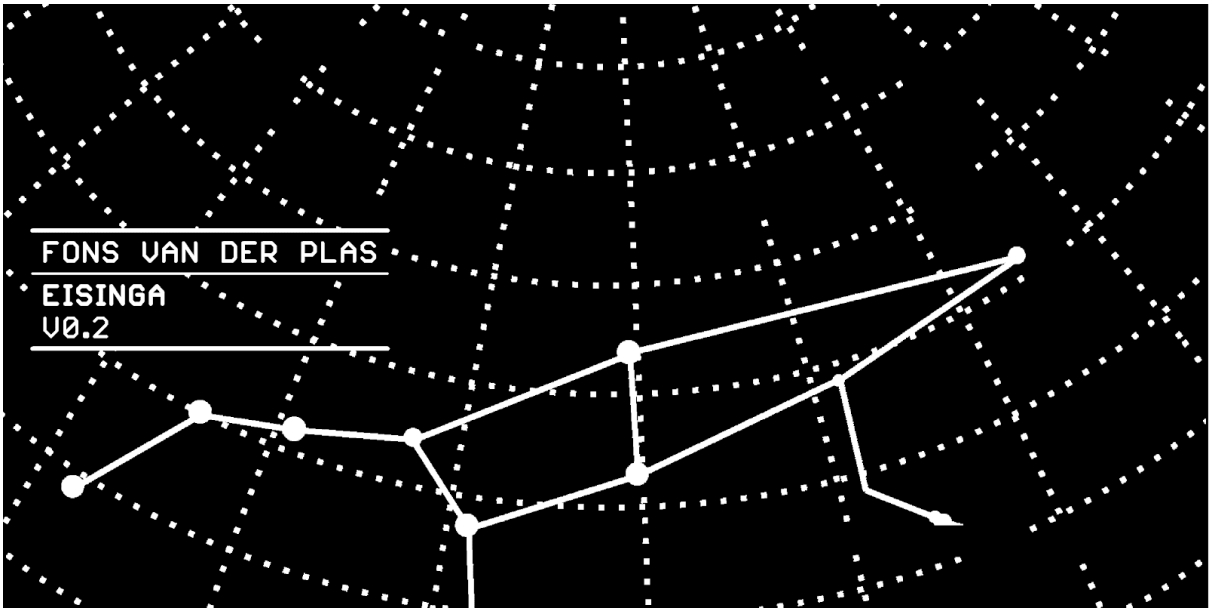


Figure 15: *The bottom silk layer, depicting the Ursa Major constellation, with the curved lines of declination and right ascension on the background. Source: own work.*

Parts

The following parts are used in the final design:

Description	Manufacturer	Name
Manufactured PCB	-	-
16x2 red LCD	-	-
Microprocessor	Atmel	ATmega328
Magnetometer	Honeywell	HMC5883L
Accelerometer	Analog Devices	ADXL354
Real time clock module	Maxim Integrated	DS1307
256 kb EEPROM	Atmel	AT24C256C
16 MHz resonator	Murata	CSTCE16M
32.768 kHz oscillator	Abracon	AB38T
Dual N-channel MOSFET transistor	Fairchild Semiconductor	FDMA1028NZ
5V regulator	Micrel	MIC5205, 5V
3.3V regulator	Micrel	MIC5205, 3.3V
Various SMD resistors and capacitors	-	-
Internal 3V battery	-	CR2032
Internal battery holder	-	CR2032, PTH
External 9V battery	-	9V
External battery clip	-	9V, PTH
On/off switch	-	-
Momentary switch	-	-
17 insulated wires	-	-

The complete list of parts, including a reference to the component in the DigiKey Electronics catalog, is included as attachment with this report.

References

Cited in this document:

1	2011. File:Azimuth-Altitude schematic.svg - Wikimedia Commons. http://commons.wikimedia.org/wiki/File:Azimuth-Altitude_schematic.svg.
2	Celestial Coordinate System. http://csep10.phys.utk.edu/astr161/lect/time/coordinates.html.
3	Timekeeping. http://csep10.phys.utk.edu/astr161/lect/time/timekeeping.html.
4	2005. Astronomical Times. http://www.cv.nrao.edu/~rfisher/Ephemerides/times.html.
5	2007. Approximate Sidereal Time - U.S. Naval Observatory ... http://aa.usno.navy.mil/faq/docs/GAST.php.
6	2003. Computing planetary positions - Paul Schlyters hemsida. http://www.stjarnhimlen.se/comp/ppcomp.html.
7	Conversion between horizontal and eq. http://star-www.st-and.ac.uk/~fv/webnotes/chapter7.htm.
8	Refraction Calculator. http://wise-obs.tau.ac.il/~eran/Wise/Util/Refraction.html.
9	2005. Yale Bright Star Catalog 5 (BSC5). http://tdc-www.harvard.edu/catalogs/bsc5.html.
10	2010. Origin of the modern constellation boundaries - Ian Ridpath. http://www.ianridpath.com/boundaries.htm.
11	2013. The Constellations | IAU. http://www.iau.org/public/themes/constellations/.
12	2013. File:Constellations ecliptic equirectangular plot.svg ... http://commons.wikimedia.org/wiki/File:Constellations_ecliptic_equirectangular_plot.svg.
13	Ozyagcilar, Talat. 2012. Implementing a tilt-compensated eCompass using accelerometer and magnetometer sensors. *Freescale semiconductor, AN* 4248.
14	Ozyagcilar, Talat. "Calibrating an ecompass in the presence of hard and soft-iron interference." *Freescale Semiconductor Ltd* (2012).
15	2013. Using EAGLE: Board Layout - learn.sparkfun.com. https://learn.sparkfun.com/tutorials/using-eagle-board-layout/routing-the-board.

Component datasheets:

Part	Link
ADXL345	http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf
HMC5883L	http://www51.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/HMC5883L_3-Axis_Digital_Compass_IC.pdf
DS1307	http://datasheets.maximintegrated.com/en/ds/DS1307.pdf
AT24C256C	http://www.atmel.com/Images/Atmel-8568-SEEPROM-AT24C256C-Datasheet.pdf
ATmega328	http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Summary.pdf
MIC5205	http://www.micrel.com/_PDF/mic5205.pdf
AB38T	http://www.abracon.com/Resonators/AB38T.pdf
CSTCE16M	http://www.murata.com/~/media/webrenewal/support/library/catalog/products/timingdevice/ceralock/p16e.ashx
FDMA1028NZ	https://www.fairchildsemi.com/datasheets/FD/FDMA1028NZ.pdf
GDM1602K	https://www.sparkfun.com/datasheets/LCD/GDM1602K-Extended.pdf

Used programs/projects:

Arduino: http://www.arduino.cc/
Arduino CMake: https://github.com/queezythegreat/arduino-cmake
I²Cdevlib: http://www.i2cdevlib.com/
MultiWii: http://www.multiwii.com/
The Mono C# runtime: http://www.mono-project.com/
GitHub: https://github.com/
StackOverflow: http://stackoverflow.com/
Sparkfun: https://learn.sparkfun.com/
Paint.NET: http://www.getpaint.net/
EAGLE: http://www.cadsoftusa.com/
Linux Ubuntu: http://www.ubuntu.com/
Monodevelop: http://www.monodevelop.com/
Google Docs: http://www.google.com/docs/about/

Attachments

The following files and documents should be included with this report:

- PCB design files (EAGLE)
- PCB manufacturing files (Gerber)
- Part list
- Presentation
- Eisinga source code
- Related source code
    - Serial to EEPROM uploader
        - Arduino code (EEPROM device)
        - C# code (host machine)
    - Sensor data interpreter (meant for debugging)
    - Constellation database generator
    - Generated constellation database (.dat file)