**Raspberry Pi Based Temperature and Weather Station**

**Gligor Bogdan-Mihai**

Student at the Technical University of Cluj-Napoca, Computer Science, Group 30414

E-mail: gligor.bogdann@gmail.com

**Abstract.** This article will present and explain how we can use a Raspberry Pi Zero W and some additional components to create a temperature and weather station. The gadget will use a 3-tone e-ink paper display because of it’s low power consumption and clarity, and the BMP180 temperature and pressure sensor for the measurements. It will also use the internet for some additional information.

1. **Introduction**

The main purpose of this project is to create a very simple and functional temperature and weather station using inexpensive and accessible components. The Raspberry Pi Zero W is a very inexpensive and powerful computer for its size and power consumption. One of the main goals of the project is to create as much functionality as possible while also staying in the limitations of the hardware used.

The final product should be a standalone and cloud connected device with a varied degree of uses.

1. **Components, circuitry and conditions**

*2.1 Components*

The following components are going to be needed:

* Raspberry Pi Zero W
* Inky pHat 2.11’’ Inch EDP Screen
* BMP180 Temperature/Pressure Sensor
* 4 Breadboard Jumper Wires (min. 10 cm length)

*2.2 Circuitry*

One of the biggest advantages of the Raspberry Pi is that it comes with a set of 40 GPIO (General Purpose Input Output) pins. They have the configuration shown below in **Figure 1**. For this particular project only 8 of the pins will be used (GPIO 2,3,8,10,11,17,22,27). An additional 2 pins are needed to power the components. As there are a multitude of power and ground pins available, the exact pins used is not relevant. It is worth noting that the Inky needs both 3.3V and 5V to function.

As shown in **Figure 2**, all 8 highlighted pins are used by the Inky pHat. The screen is called a “pHat” because it resembles a hat in the sense that it fits perfectly over the Raspi’s pins. No soldering or breadboard is required to mount it. Using an already existing Python library the screen can be easily controlled.

To see how the Inky pHat sits on the Pi check **Figure 11.3** at the end of the article.

In addition to the Inky pHat using these 8 pins, 2 of them are also used by the BMP180 sensor. GPIO 2 and 3 are used by the sensor to send the data to the Pi after it was registered. These pins can be used at the same time between the two different components by not triggering them simultaneously. Since the Inky already covers pins 2 and 3 it is necessary to solder the wires to these pins to the underside of the pins. It is also necessary to solder any ground and 3.3V pins for power, since these are also covered by the Inky. The Vin is soldered to any 3.3V pin, the GND to any ground pin, the SDA to GPIO pin 2, and the SCL to GPIO pin 3. The circuitry should be complete at this point.

|  |
| --- |
|  |
| **Figure 1.** The diagram for the Raspberry Pi’s GPIO pins |

|  |
| --- |
|  |
| **Figure 2.** The highlighted pins are used by the Inky pHat |

|  |
| --- |
|  |
| **Figure 3.** The pinout of the BMP180 |

*2.3 Conditions*

The gadget is to be stored indoors. It is not waterproof nor does it resist in extreme weather conditions. While this could be solved by using a special case, it defeats the purpose of the project. It is meant to be used indoors to monitor the temperature inside the room, while also using the internet for getting the outside weather conditions.

1. **Coding, scripting and graphics**

*3.1 Coding*

I decided to use Python 3 (and Python 2) as the coding and scripting language of the project. Since both the Inky pHat and the BMP180 have libraries written in Python, and the language works very well on the Pi, it was a no brainer.

To control the Raspberry Pi remotely I used Putty to SSH into it from my laptop. It gave me access to a terminal open inside the Pi’s Raspbian (Linux) Operating System, which was more than enough to work with.

The bulk of the code in this project is found in the file “term.py”. The main objective of this program is to unite all the information and graphics and display them on the Inky.

It uses numerous reading functions (ex. **Figure 4**) to read from separate .txt files information that was produced by the other 2 programs working in parallel to it. It also has file writing functions for error handling reasons.

|  |
| --- |
|  |
| **Figure 4.** Example of a read function |

The biggest function is by far the thermometer() function. While not quite appropriately named, its only parameter is a date parameter. It finds out the weekday from the parameter, which it will display on the screen. Here it also extracts the information gathered by the sensor, like the temperature, atmospheric pressure, and altitude (calculated using the pressure) and displays it also.

The information on the screen is updated every 15 minutes. While this might seem a little long, it greatly decreases power consumption and it diminishes burning (explained below).

To display the information the InkyPhat library is used. It has functions that read both text and image files as input. In the end it puts all of them together to create a coherent and aesthetically pleasing graphics. This will be further expanded on in the scripting part of the explanation.

There is also a function that clears the screen. As the screen is an e-Ink display it is prone to burning (old text is present, faded, even after new text appears). To eliminate this problem, every 5 cycles of updating the information on the screen, the function will be triggered to completely flush the screen and remove any burning marks.

See **Figure 5** for the extraction part, and **Figure 6** for the displaying part.

|  |
| --- |
|  |
| **Figure 5.** Code responsible for extracting the required information |

|  |
| --- |
|  |
| **Figure 6.** Code responsible for the graphics shown on the Inky pHat |

*3.2 Scripting*

You might ask why this category is not included in the coding part. I have decided that all information gathering programs should be called scripts and be spoken of separately.

The first script is a web scraper called “scraper.py” that sends a request every 5 minutes to a weather information website ( <https://www.vremea.ro/cluj/clujnapoca/> ), asking for the weather in the city of Cluj-Napoca, where I live. The city can be changed within the source code if needed. After that using some functions found in the bs4 web scraping library, it extracts the current weather in the form of a string of text composed of keywords. An example of an output of this script would be “partial innorat”. It then writes these keywords into “vreme.txt”, where they can be read by the big “termp.py” program. Here a dictionary is used to assign these keywords the corresponding weather icon. If the keyword is not found in the dictionary, then it is written in the mistakes.txt file for debugging.

See **Figure 7** and **Figure 8** for the code, and **Figure 9** for the weather icons.

|  |
| --- |
|  |
| **Figure 7.** The dictionary responsible for assigning the keywords the right icons |

|  |
| --- |
|  |
| **Figure 8.** The web scraper script |

|  |
| --- |
|  |
| **Figure 9.** Icons used to represent the weather |

The other script used is “bmp180-example”. It is quite a weird file, since it does not include a .py extension at the end. It is also written in Python 2, not Python 3, like the rest of the programs. It was necessary to use this file as the library used to control the BMP180 sensor is old and very hard to get working. What is important is that it does work, and it writes the temperature, pressure, and altitude in the files “temp.txt”, “press.txt” and “alt.txt” where they can be read by the big “term.py” program. It does this every 15 seconds.

See **Figure 10** for the code.

|  |
| --- |
|  |
| **Figure 10.** The sensor reading script |

*3.3 Graphics*

The icons are created by me in Photoshop, taking some inspiration from 8-bit style games. The font is not, and it’s included in the Inky pHat library. It’s called AmaticSCBold, and I use it at 20 pt.

1. **Conclusion**

In conclusion, having presented all the necessary information to create such a gadget, I think I reached the goal of this article. I hope whoever reads this paper has a better understanding of how to use the Raspberry Pi Zero W in projects such as the one presented. I also hope I inspired people to take on this project and hopefully build and improve on it. In **Figure 11.1**, **Figure 11.2** and **Figure 11.3** the final product can be seen. It can also be observed that I added an external battery to the gadget to make it portable.

|  |
| --- |
|  |
| **Figure 11.1.** The final form of the project |

|  |
| --- |
|  |
| **Figure 11.2.** Upper view of the final form of the project |

|  |
| --- |
|  |
| **Figure 11.3.** Bottom view of the final form of the project |

1. **References**

(Images)

1. <https://pinout.xyz/pinout/inky_phat>, 2.<http://acoptex.com/project/3908/raspberry-basics-project-19a-raspberry-pi-zero-w-board-setup-at-acoptexcom/>