

**Embedded System Project**

Baby Room Thermometer

**Date**: 21/12/2023

**Supervisor**: Dr. Salman Alaraibi

**Done By**: Dana Barhoom

**ID**: A00200

**The Challenge:**

For newborn babies, it is generally recommended that their environment be maintained and not go below 20 degrees Celsius. This is usually an issue for parents since sometimes guardians are unaware that the temperature drops below what is desired for their baby. In addition, sometimes parents have to leave the room once their child rests and thus are unaware if the temperature is dropping. To solve this issue, the parent must know that the temperature of the room doesn’t drop below 20 degrees, continuously to be able to regulate it for their child.

**The Solution:**

The goal of this product is to assist parents in maintaining a suitable and comfortable environment for their newborns. A solution to this problem would be having a room thermometer designed aesthetically for babies to monitor and display the room temperature accurately and continuously. This room thermometer would show the temperature reading of the room in degrees Celsius. It will also have a built-in alert system to notify the parent if the temperature drops. Without the alert system, the parents would not know that the temperature dropped unless they check the system reading. The alert system combines both light in the form of LEDs and sound for auditory notification. This is particularly helpful for adults, especially when they are not present in the room the child is in to be alerted via sound if the temperature drops. This product would be powered by a plug so that it would always be ON when the child is in a specific room. This can help parents ensure that the temperature is not too low, as temperatures below 20 degrees Celsius might not be optimal for newborns. This product is a solution to the challenge addressed earlier as it is a:

Room Thermometer with a playful design for babies, that also has the additional feature to produce light and sound to notify and alert adults once the temperature drops to a certain limit.

**System Block Design:**

****

**Figure 1: The block design of the system illustrates the relationships between the elements involved.**

Figure 1 shows how the temperature sensor is the input signal present in the Raspberry Pi Pico microprocessor in channel 4 through ADC. The processing is completed in the Raspberry Pi Pico microprocessor. The result after the processing is three outputs as shown above. The LED, buzzer, and OLED.

The temperature reading is displayed on the OLED, which is an output. The other output responses are the LED being lit for light and the buzzer turning on for sound. These two outputs occur only once the temperature drops below 22 for the baby to stay in an ideal environment, which is provided by the temperature sensor input. These outputs are there to alarm parents or guardians about the temperature change.

The product would be powered by being plugged into a socket to monitor the room's temperature continuously without being interrupted by having to be charged.

**Hardware and Materials:**

**Table 1 Hardware items:**

|  |  |  |
| --- | --- | --- |
| **Hardware** | **Explanation** | **Picture** |
| Raspberry Pi Pico microprocessor | The processing unit for the input and outputs. | A close-up of a circuit board  Description automatically generated |
| OLED | Hardware content needed to display the temperature reading of the room for the user to know what the temperature is at any time. |
| Built-in Temperature Sensor | The sensor that will be used to detect changes in temperature and display them on the OLED |
| LED | Hardware content needed to display light for the user to observe. |
| Buzzer | Hardware content needed to provide sound as a method to notify individuals. |
| USB | For the Raspberry Pi Pico. |  |

* **Product materials in the next page:**

**Product materials:**

Plastic Casing is the chosen type of casing or covering of the input and outputs of the product that will house the electronic hardware mentioned in Table 1. This is because it can be easily molded to provide an eye-catching design for the thermometer to be appealing for customers to buy. In addition, plastic casing is considered the best option as it is durable and relatively inexpensive, and affordable. This would make the product more appealing to guardians.



Figure 2: home thermometers using a plastic casing over its display [1].

Figure 2 shows how most home thermometers use a plastic casing around their temperature display, as it is durable and has an attractive look.



Figure 3: shows a baby home thermometer using a plastic casing with an appealing design suitable for children [2].

In Figure 3, this product’s concept design is considered more attractive to parents and guardians for their children as it has a design that is appealing and is more designated to kids. This project’s thermometer is inspired by Figure 3’s product design element. This product also uses a plastic casing but with a different design element than Figure 2.

* **Powering:**

In this product, we are using a Raspberry Pi microprocessor, which is designed to be powered by a micro-USB cable connected to a power source, which can be a traditional AC power outlet. To power this product, it has to be plugged into a socket to provide energy for this solution. This is why, it requires the Raspberry Pi micro-USB to be connected to a USB charger, which will also be plugged into the AC power outlet to power the solution, similar to what was shown in figure 4 below.



Figure 4: Baby home thermometer powering method [3].

**Interface:**

**There are two interfaces used to interact with the user. They are:**

1. Digital Display:

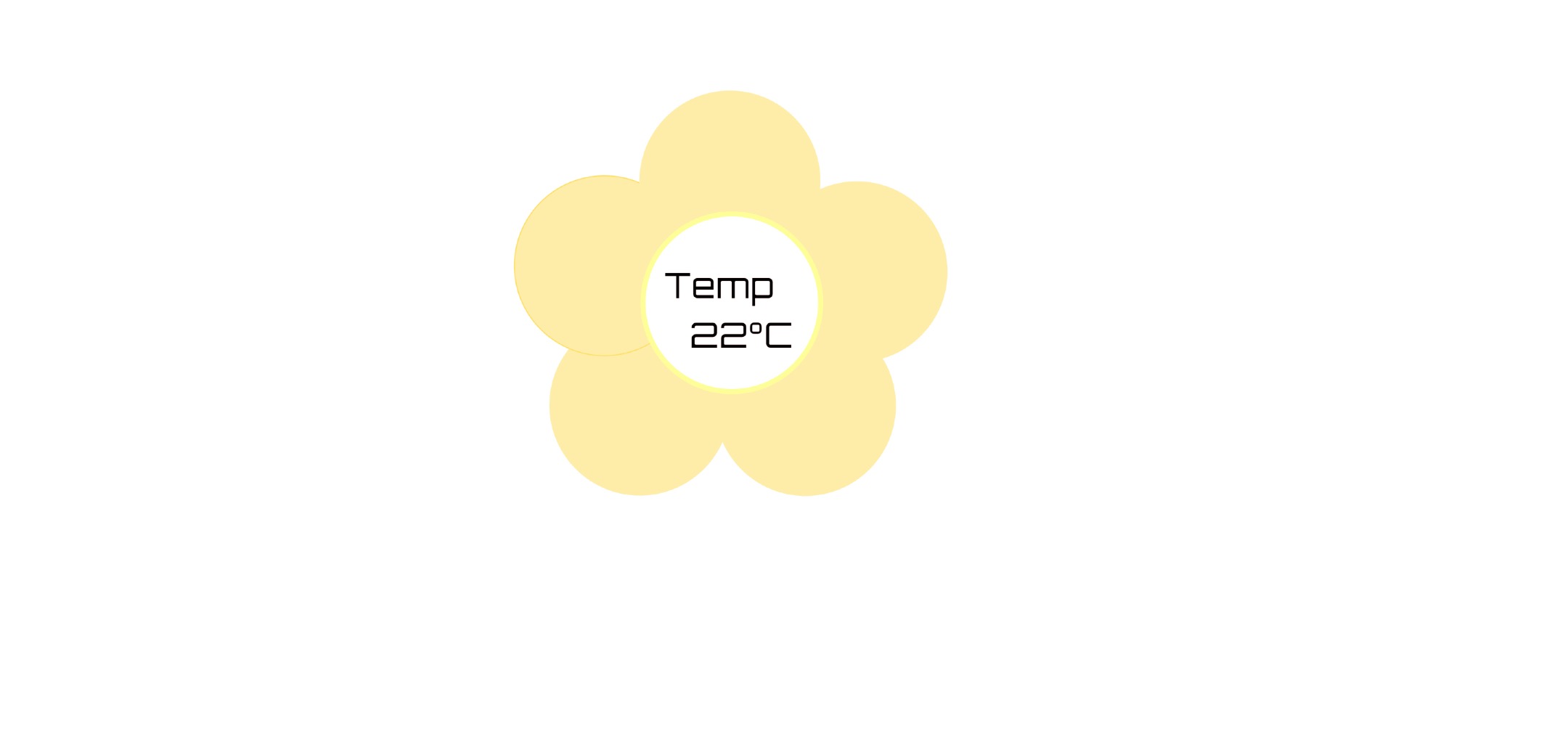
It is the numerical digital display of the temperature reading from the temperature sensor. The display is presented on an OLED screen that tells the user what the temperature of the room is.

1. Alert System:

This home thermometer has an alert system that notifies the user when a specific temperature threshold is reached. This is done by means of audible alarms done by the buzzer, and visual alerts which are done by the LED light present in the solution.

**Prototype Sketch:**

Prototype sketch done using Sketchpad.



* The design is made in a flower shape to be placed anywhere around the room.
* The flower shape is going to be the housing of the electronic parts which will be placed inside the plastic flower shown in the design above.
* The design is more appealing to babies because of the baby color flower casing around the thermometer reading.
* The cute-looking element of the product makes it desirable to be bought by parents for their child to be placed in their room or near the crib. It would look like a decorative toy instead of a thermometer, which makes it a beautiful product to buy.

**Implementation:**

The step-by-step sequence of events:

1. An important output required is the OLED display. The code for this must be done that would allow the numerical reading of the internal temperature sensor to be displayed on the OLED. The code for this must be downloaded first.
2. After this the code for the alert system output of the LED and buzzer once the temperature is lower than 22 degrees is done. Here, We must import the machine and sleep to make use of the appropriate pins present in the Raspberry Pi microprocessor.
3. The input is received from the temperature sensor. Since the system is based on temperature readings from the temperature sensor, importing the necessary module ADC in channel 4 is required. This must be the first thing to be declared. After this, declaring the output pins for the LED and the buzzer is also necessary.
4. After this for the temperature sensor, set the conversion factor to read the value from the sensor. Then we must convert that value to be read numerically and then display it. The value will be displayed in the output OLED which was coded first.
5. The other outputs to notify the user, are the LED and buzzer which must be linked with the sensor reading. This is done using a statement that would cause the LED and the buzzer to turn on once the temperature reading is less than 22 since for babies.
6. Combining this logic to create the solution to our problem using the following code developed and shown in the next section.

**Micropython Code:**

**The micropython code used to create the solution to the problem:**

1. **For the OLED we need to download the following code:**

from micropython import const

import framebuf

SET\_CONTRAST = const(0x81)

SET\_ENTIRE\_ON = const(0xA4)

SET\_NORM\_INV = const(0xA6)

SET\_DISP = const(0xAE)

SET\_MEM\_ADDR = const(0x20)

SET\_COL\_ADDR = const(0x21)

SET\_PAGE\_ADDR = const(0x22)

SET\_DISP\_START\_LINE = const(0x40)

SET\_SEG\_REMAP = const(0xA0)

SET\_MUX\_RATIO = const(0xA8)

SET\_COM\_OUT\_DIR = const(0xC0)

SET\_DISP\_OFFSET = const(0xD3)

SET\_COM\_PIN\_CFG = const(0xDA)

SET\_DISP\_CLK\_DIV = const(0xD5)

SET\_PRECHARGE = const(0xD9)

SET\_VCOM\_DESEL = const(0xDB)

SET\_CHARGE\_PUMP = const(0x8D)

**class** SSD1306(framebuf.FrameBuffer):

    def \_\_init\_\_(**self**, width, height, external\_vcc):

**self**.width = width

**self**.height = height

**self**.external\_vcc = external\_vcc

**self**.pages = **self**.height *// 8*

**self**.buffer = bytearray(**self**.pages \* **self**.width)

**super**().\_\_init\_\_(**self**.buffer, **self**.width, **self**.height, framebuf.MONO\_VLSB)

**self**.init\_display()

    def init\_display(**self**):

**for** cmd **in** (

            SET\_DISP | 0x00,

            SET\_MEM\_ADDR,

            0x00,

            SET\_DISP\_START\_LINE | 0x00,

            SET\_SEG\_REMAP | 0x01,

            SET\_MUX\_RATIO,

**self**.height - 1,

            SET\_COM\_OUT\_DIR | 0x08

            SET\_DISP\_OFFSET,

            0x00,

            SET\_COM\_PIN\_CFG,

            0x02 **if** **self**.width > 2 \* **self**.height **else** 0x12,

            SET\_DISP\_CLK\_DIV,

            0x80,

            SET\_PRECHARGE,

            0x22 **if** **self**.external\_vcc **else** 0xF1,

            SET\_VCOM\_DESEL,

            0x30,

            SET\_CONTRAST,

            0xFF,

            SET\_ENTIRE\_ON,

            SET\_NORM\_INV,

            SET\_CHARGE\_PUMP,

            0x10 **if** **self**.external\_vcc **else** 0x14,

            SET\_DISP | 0x01,

**self**.write\_cmd(cmd)

**self**.fill(0)

**self**.show()

    def poweroff(**self**):

**self**.write\_cmd(SET\_DISP | 0x00)

    def poweron(**self**):

**self**.write\_cmd(SET\_DISP | 0x01)

    def contrast(**self**, contrast):

**self**.write\_cmd(SET\_CONTRAST)

**self**.write\_cmd(contrast)

    def invert(**self**, invert):

**self**.write\_cmd(SET\_NORM\_INV | (invert & 1))

    def show(**self**):

        x0 = 0

        x1 = **self**.width - 1

**if** **self**.width == 64:

            x0 += 32

            x1 += 32

**self**.write\_cmd(SET\_COL\_ADDR)

**self**.write\_cmd(x0)

**self**.write\_cmd(x1)

**self**.write\_cmd(SET\_PAGE\_ADDR)

**self**.write\_cmd(0)

**self**.write\_cmd(**self**.pages - 1)

**self**.write\_data(**self**.buffer)

**class** SSD1306\_I2C(SSD1306):

    def \_\_init\_\_(**self**, width, height, i2c, addr=0x3C, external\_vcc=**False**):

**self**.i2c = i2c

**self**.addr = addr

**self**.temp = bytearray(2)

**self**.write\_list = [b"\x40", None]

**super**().\_\_init\_\_(width, height, external\_vcc)

    def write\_cmd(**self**, cmd):

**self**.temp[0] = 0x80

**self**.temp[1] = cmd

**self**.i2c.writeto(**self**.addr, **self**.temp)

    def write\_data(**self**, buf):

**self**.write\_list[1] = buf

**self**.i2c.writevto(**self**.addr, **self**.write\_list)

**class** SSD1306\_SPI(SSD1306):

    def \_\_init\_\_(**self**, width, height, spi, dc, res, cs, external\_vcc=**False**):

**self**.rate = 10 \* 1024 \* 1024

        dc.init(dc.OUT, value=0)

        res.init(res.OUT, value=0)

        cs.init(cs.OUT, value=1)

**self**.spi = spi

**self**.dc = dc

**self**.res = res

**self**.cs = cs

        import time

**self**.res(1)

        time.sleep\_ms(1)

**self**.res(0)

        time.sleep\_ms(10)

**self**.res(1)

**super**().\_\_init\_\_(width, height, external\_vcc)

    def write\_cmd(**self**, cmd):

**self**.spi.init(baudrate=**self**.rate, polarity=0, phase=0)

**self**.cs(1)

**self**.dc(0)

**self**.cs(0)

**self**.spi.write(bytearray([cmd]))

**self**.cs(1)

    def write\_data(**self**, buf):

**self**.spi.init(baudrate=**self**.rate, polarity=0, phase=0)

**self**.cs(1)

**self**.dc(1)

**self**.cs(0)

**self**.spi.write(buf)

**self**.cs(1)

1. **The code for the OLED displays the reading of the sensor and the alert system output of the LED and buzzer once the temperature is lower than 22 degrees:**

from machine import Pin, I2C

from ssd1306 import SSD1306\_I2C

import framebuf

import machine

import utime

sensor\_temp = machine.ADC(4)

conversion\_factor = 3.3 / (65535)

LED = Pin(16, Pin.OUT)

buzzer = Pin(15, Pin.OUT)

WIDTH  = 128

HEIGHT = 64

i2c = I2C(0, scl=Pin(9), sda=Pin(8), freq=200000)

print("I2C Address      : "+hex(i2c.scan()[0]).upper())

print("I2C Configuration: "+str(i2c))

oled = SSD1306\_I2C(WIDTH, HEIGHT, i2c)

**while** **True**:

    reading = sensor\_temp.read\_u16() \* conversion\_factor

temperature = 27 – (reading – 0.706)/0.001721

print (temperature)

    oled.fill(0)

oled.blit(fb, 96, 0)

oled.text("Temp: ",6,8)

oled.text(str(round(temperature,2)),50,8)

oled.text(("°C ",,95,8)

utime.sleep(2)

    oled.show()

if temperature ‹ 22:

buzzer.high( )

LED.high()

utime.sleep (1)

elif temperature › 22:

LED.low()

buzzer.low( )

utime.sleep (1)

else:

print(“error”)

**Testing:**

**Test case 1:**

Testing the ability of the OLED to display the correct numerical value read by the temperature sensor input. This is to test whether the OLED code works with the temperature sensor and displays the desired output with the correct numerical values in degrees Celsius. This was the first to be tested to be able to see whether the Raspberry Pi microprocessor can read input from the temperature sensor and display the OLED output.

After much trial and error and finding the right code combination to show the temperature in degrees Celsius on the OLED, the result was finally successful, and from a scale of 1 to 10, the success was an 8.

**Test case 2:**

The second test case was testing the LED’s ability to blink successfully when the reading of the temperature input was lower than 22 and incorporating the buzzer sound once the temperature dropped. This test was done excluding the OLED’s reading. This is to confirm the code for the LED and Buzzer functionality without the OLED to avoid disrupting any of the signals produced by these outputs. Testing those two outputs together as temperature drops were successful. And from a scale of 1 to 10, the success rate was a 10.

**Test case 3:**

The third test case scenario included testing the coding success of the three outputs combined with the input temperature sensor. This trial tested all three outputs, the OLED temperature reading, LED, and buzzer sound together when the temperature drops. This was the most difficult part of the project as this test required the greatest work. This is because the appropriate combination of code must be found that allows the three outputs to work together with the temperature sensor input. After long hours of trial and error, the right combination of code was found and relatively successful. On a scale of 1 to 10, the success rate was a 7.

**Conclusion and Future Work:**

This project involved the use of a Raspberry Pi temperature sensor input to produce output sensor reading on OLED, in addition to buzzer sound and light. This is used to create a product that would be helpful for parents and adults who have children and need the temperature of the room to be regulated. I have learned several things from this project. Firstly, I gained knowledge about the Raspberry Pi microprocessor. I learned how to read temperature values from the sensor and process them with micropython. In addition, I familiarized myself with creating outputs based on the temperature readings, such as displaying the values on an OLED and creating sounds and light through the use of LED and Buzzer.

One difficulty I encountered during the project was troubleshooting and incorporating the temperature sensor reading with the blinking of the LED and the buzzer sound. It was important to me that all of these outputs worked together for the product to be successful. After hours of searching for the solution to this problem, I was able to accomplish my goal with minimal errors.

For future improvements, further practice with micropython using several inputs and outputs must be done to enhance the quality, accuracy, and success of the project or any future projects done using Raspberry Pi Microprocessor. This is to improve the skill of working with microprocessors with projects that require several inputs and outputs to be incorporated together for a certain purpose.

**Citations:**

[1] Bodi-Tek, “Digital Room Thermometer and hygrometer,” Bodi, https://www.bodi-tek.com/products/room-thermometer-and-hygrometer (accessed Dec. 23, 2023).

[2] “Purflo Starlight Colour Changing Baby Room Thermometer: USB rechargeable digital temperature monitor for indoor & Nursery: One of Your baby essentials,” Amazon.co.uk: Baby Products, https://www.amazon.co.uk/Purflo-Starlight-Changing-Temperature-Monitor/dp/B0763RQBX9 (accessed Dec. 23, 2023).

[3] “Purflo Starlight Colour Changing Baby Room Thermometer: USB rechargeable digital temperature monitor for indoor & Nursery: One of Your baby essentials,” Amazon.co.uk: Baby Products, https://www.amazon.co.uk/Purflo-Starlight-Changing-Temperature-Monitor/dp/B0763RQBX9 (accessed Dec. 23, 2023).