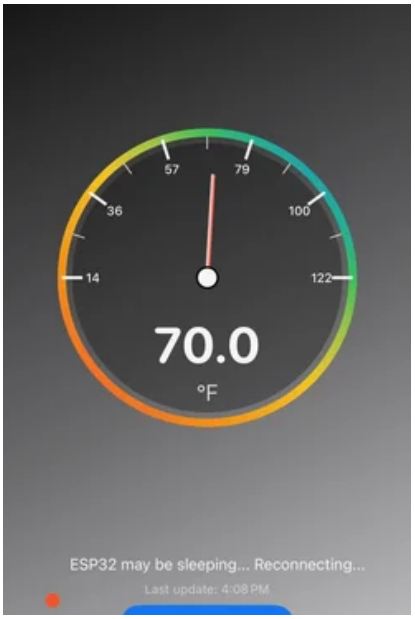
Kelvyn--A Direct Temperature Sensor for the IPhone

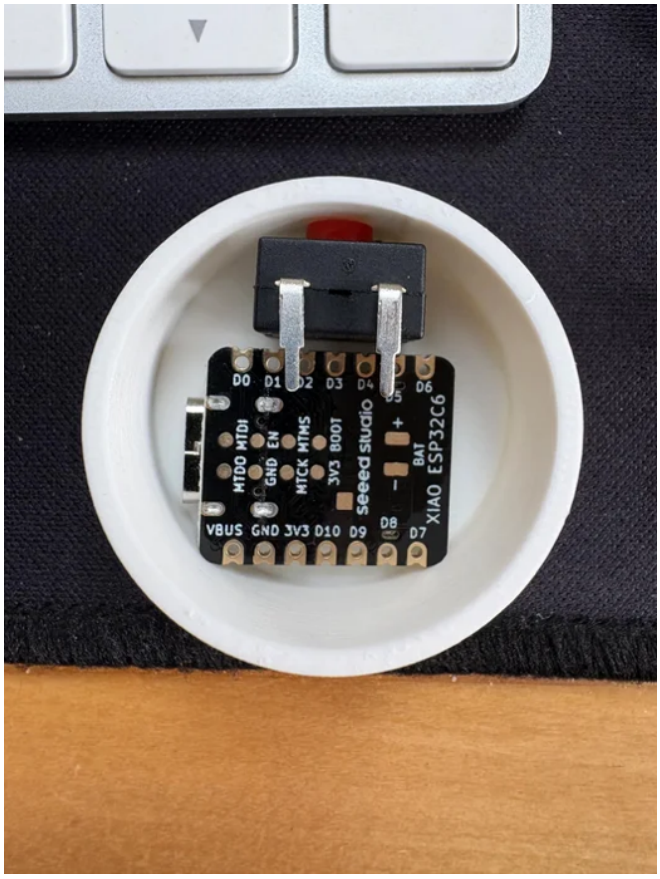
By rabbitcreek in CircuitsMicrocontrollers
Unpublished

Introduction: Kelvyn--A Direct Temperature Sensor for the IPhone



Living in Alaska the first question you ask most mornings is what temp is it? It's not like Hawaii where you know what you're going to be wearing every day. The temperature can of course vary from terminally below zero to (now) in the high 90's. Temperature sensors are now ubiquitous and most circuit boards have a couple casually placed into hidden crevices. The neat App: phyphox teases quite a few out for experimentation but one prominently missing is Temperature. So I built a little open source temperature sensor that you can casually add to your phone. It connects with Bluetooth and comes with a free App that I built so you can monitor your temperature anywhere you roam. The sensor is built around a Xiao ESP32C6 and a tiny battery. It sleeps most of the time and wakes up advertises the temp and then sleeps. It uses a DS18B20 with its usual conventions--accurate to +- 0.5F. It costs about $10 to make and 3D print and weighs 1/2 oz. Since it spoofs a heart rate monitor for its Bluetooth connection it also pairs very easily with Strava in case you're more interested in the temperature than your pulse.

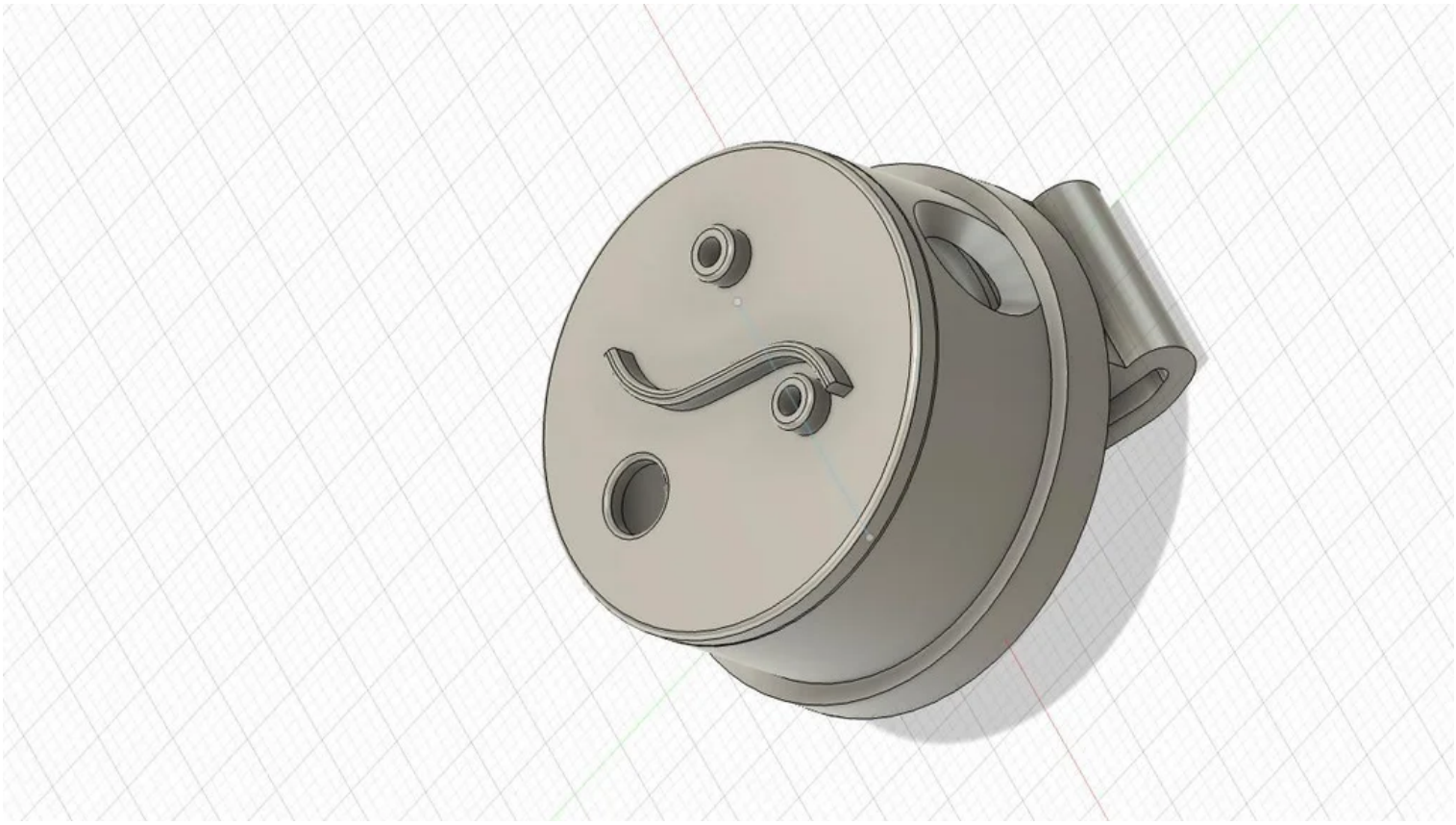
Supplies



Really, there's not much to this project. A few simple supplies:

1. Xiao ESP32C6 $6
2. 100mah Lipo Battery $3
3. Switch ON/OFF Adafruit #1683
4. very tiny LED's ---find these on Amazon--they are about the size of sand grains! (Pre-wired micro-LED)
5. DS18B20 -- $1

Step 1: Print Your Parts



All parts were printed with PLA-Tough from Bambu Labs. Im not sure actually how tough this stuff is but since this will be exposed to higher outdoor temps you might want to print them in PET-G. Supports were used for most of the parts. The two main ones are just for the case and the others are my attempts at providing attachment point for the device. You can also just provide and easy glue-on pin that would also work well.

Step 2: Wire It

This is fairly simple to wire. The temp probe has three wires to it as you can see in the diagram above. Power is connected to D10 on the ESP32...this will allow you to shut down power when it goes to sleep. Ground to ground and Data (center pin) to D7. This has to be held high so also connect a resistor (4.5k ohms) between D7 and D10 so it keeps the data line high during readings. D8 and D9 are connected to the positive side of the two micro LEDs and their ground lines are connected through a small resistor to the ground. The battery is simply connected to the battery terminals on the back of the ESP through a switch on the Pos side. That is it for the wiring.

Step 3: Building It

This thing is pretty tiny so you have to really cram in these parts. The sensor is bent from its connections at a 90 degree angle and its head is placed through the small hole for it in the cap. It is hot-glued and sealed into its position. The two micro-LEDs are also poked into the tiny holes in the top for them and sealed with hot glue. The ESP is turned over and placed through the opening in the lower body and the battery is placed on its back. You might want to hot glue these in so they don't move. The ON/OFF switch is placed within its housing. All wiring connections are done at this point and the upper part of the case is sealed with superglue. Make sure you test all wiring and connections before sealing it up.

Step 4: Programming It

This is fairly straightforward piece of coding. The Bluetooth connection is based on my prior work on VO2 max unit that uses Andress Spiess work in YouTube #174. It uses the bluetooth credentials from a heart rate monitor and this allows the software to pare up with an Swift App of your own design and also to spoof the Strava App into utilizing the data for its heart rate monitor and graph the results even on the free version....if your into that sort of thing. The one wire bus is connected to D7 and the LED-Right is connected to D8 and LED-Left is connected to D9. Bluetooth is initialized. Fllash function is used to flash the results of the temperature reading into the two tiny leds. The tens position is left-Red and the ones position is Right-green and it flashes out once/sec. The setup() function sets up the led's and the power to the sensor as well as initiating the sensor. Standard Dallas temp does not work on the C6 yet so you have to use the modified code as written or it won't compile. This unit waits for a connection and then times out after 30 seconds if there is no pickup. It then goes to sleep for a variable amount of time to save battery life...timeOut has been set to measure every ten minutes.

The Swift App that I wrote to accompany this unit will be open-sourced and available on Github as soon as the App is (hopefully) approved.

Step 5: Using It

When available, download the App. Push the button at the top of the unit and it will immediately pair with the App...you can also push the pair button if it has not been used before. Don't pair the unit in your Bluetooth settings under System on your phone it only tends to screw up pairing. If this accidentally happens push "forget this device" and it should come back online. The first screen can be toggled for F/C and will tell you the last time it has paired. The second screen has a graph with the accumulated data. It can also be scaled for day/week/year. It also has high/low and averages. For Strava connection make sure it is listed under "sensors" and listed as connected. You have to make sure the Kelvyn App is turned off or it will compete with the Strava App. To activate reading push the record button and you should see the heart icon spill data. At the close of your race you will see a graph of your temperature over distance. The unit is recharged through the USB-C port but remember to turn the button to the on position. You can tell if the button is on by noting the blinking LEDs which will count out the current temperature readings...for 72...7 red blinks and 2 green blinks. This minimal output saves the need for a screen and diminishes battery usage.