Sip Log: A Mobile Water Intake Monitor and Reminder

Hanna Suzuki

Bedford High School
Bedford, MA, 01730, USA

Abstract

Sip Log is a mobile device that computerizes a water bottle to monitor water intake during physical activity. Attached to the bottom of a bottle, it keeps track of water intakes, records intake logs in the cloud, helps estimate water consumption, and issues hydration reminders via smartphone notifications. It aims to aid athletes be more aware of their water intake and prevent dehydration, which affects their condition and performance. Sip Log uses a battery-operated ESP32 microcontroller that runs CircuitPython code to detect water intake with a 3-axis accelerometer and interact with the cloud with WiFi.

X.1 Introduction

Hydration is a crucial part of our everyday lives. It is especially important when you are physically active, like athletes. Anyone who is physically active can lose up to two to three quarts of fluid per hour [1]. Athletic performance can drastically decline when athletes do not take in enough fluids to account for the fluids they lose, which ends up in dehydration [2-5]. Dehydration increases risk of injury, exhausts athletes, and decreases their mood and concentration.

Therefore, the objective of this project is to develop a handy device that monitors water intake during a sports practice/event. It is intended to aid athletes to be more aware of their water intake and prevent dehydration to ensure their best condition and performance.

The proposed device, called Sip Log, uses a battery-operated ESP32 microcontroller (M5StickC PLUS2) that is attached to the bottom of a water bottle (Figs. 1 and 2). It runs CircuitPython code to detect water intake by sensing the tilt of the bottle with a 3-axis accelerometer. Each detection is logged in the Kintone cloud database [6] to keep track of water intakes and helps estimate water consumption. Additionally, Sip Log sends out hydration reminders via smartphone notifications when no intakes are recorded for a while. It is as light as 3.5 oz (96

grams). Its battery life is 2.5 to 3 hours with its default operational settings, which is long enough for a sports practice/event.

Fig. 1 Sip Log (left) separated from a Water Bottle (right)

Fig. 2 Sip Log attached to the Bottom of a Water Bottle

X.2 Dehydration and its Effects

Dehydration occurs when the body is taking in less fluids than it is losing [2, 3]. The people who are most at risk are those being active in the heat, such as athletes in the summer weather. Although anyone who is not drinking and replenishing their body back with the fluids they need

are at risk. If hydration is not taken seriously, even for non-athletes, it could turn into something that requires medical attention.

The signs of dehydration vary from person to person, but most are subtle and overlooked. Dry mouth and tongue, headache, exhaustion, muscle weakness, dry skin, dark urine, and dizziness are all symptoms of dehydration [2, 3]. This is why it is important to not just rely on thirst and continuously hydrate. Many professional athletes have a hydration schedule they follow that covers before their match/game to after, ensuring that they execute their best and keep their bodies healthy [4, 5]. The National Federation of State High School Associations (NFHS) also states that hydration before, during, and after physical activity is integral to healthy, safe, and successful sports participation [7].

Dehydration affects the athlete's condition, performance, and even life [7]. Since water cushions the joints without enough of it, muscles will not function properly. This results in affecting the athlete's speed, flexibility, and endurance. It can also increase the risk of injury. Body fluids also moisten up the surrounding air to make breathing easier. Dehydration then makes breathing harder, exhausting the athlete quicker. This exhaustion will then cause heavy sweating, dizziness, nausea, and headaches. Which then affects the athlete's mood and concentration, preventing them from executing their best performance.

X.3 Setting Up Hardware Components

This section describes how to prepare and set up hardware components to build Sip Log.

X.3.1 Preparing Hardware

Sip Log is built with an ESP32 microcontroller and an attachment part for a water bottle.

- M5StickC PLUS2 (1x): This is equipped with an ESP32 microcontroller, an 8 MB flash memory, a WiFi module, and a LiPo battery (200 mAh). It has a Type-C USB port, three buttons, a color LCD display (ST7789v2, 1.14 inches, 135 x 240 pixels), and a 6-axis IMU (Inertial Measurement Unit; MPU6886), which combines a 3-axis accelerometer and a 3-axis gyroscope. The device weighs 17 grams. The dimensions of its enclosure are 48 x 25 x 13 mm. Amazon Standard Identification Number (ASIN): B0F3XQ22XS.

- Type-A to Type-C USB cable (1x): This is used to connect a M5StickC PLUS2 to a laptop/desktop computer where software development takes place. ASIN: B07DC5PPFV.

- Water bottle boot (1x): This is an attachment for the bottom of a water bottle. Sip Log places a M5StickC PLUS2 in this attachment (Figs. 1 and 3). Make sure that its diameter

matches the bottle's diameter and its size is larger than the dimensions of a M5StickC PLUS2. ASIN: B0BW97RSMG. This project uses a 32oz Wide Mouth Hydro Flask bottle. Its diameter is 9cm. The boot's depth is 4.8cm. Since this is much longer than the height of a M5StickC PLUS2, this project places a styrofoam base in the boot and tapes the M5StickC to the base (Fig. 3). This helps stabilize the device in the boot.

ASINs can be used for product searches at **`https://www.amazon.com`**.

Fig. 3 Styrofoam Base placed in a Bottle Boot to Stabilize an M5StickC PLUS2

X.3.2 Installing a USB Driver

The first step to use a M5StickC PLUS2 is to install a USB driver (CH9102) on a laptop/desktop where software development takes place. Download the driver from [8] and follow the instructions in [9].

Then, plug a USB cable to the M5StickC's USB port (Fig. 3) and connect the cable to the laptop/desktop computer. If the M5StickC is turned off, this turns it on and boots it. Otherwise, the device re-boots automatically. While connected to the computer, it charges the M5StickC's battery. To turn off the device, press the power button (Button C; Fig. 3) for six seconds.

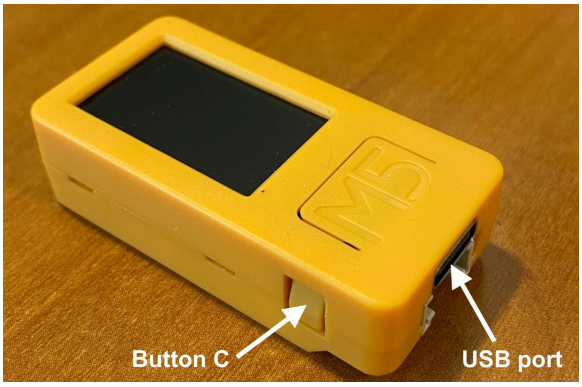


Fig. 4 USB Port and Power Button (Button C) on an M5StickC PLUS2

X.3.3 Installing CircuitPython Firmware

Sip Log uses CircuitPython [10] for software development. Install CircuitPython firmware onto the M5StickC with a web browser on the laptop/desktop computer. Visit [11] with the browser and click the "Open Installer" button. This project uses CircuitPython 9.x (9.2.8), not 10.x. This project also uses Thonny [12] for coding using the REPL prompt, but any CircuitPython-compatible tools should work as well.

The libraries available in the CircuitPython firmware are listed at [11]. They can also be printed on the REPL prompt in CircuitPython by running `help("module")`.

X.3.4 Configuring WiFi Credentials

M5StickC PLUS2 has a WiFi module, and this project uses it to record water intake logs in the cloud. To set up WiFi credentials, find `settings.toml` in your CircuitPython file system and add the following two lines in there.

```
CIRCUITPY_WIFI_SSID = "abc"
CIRCUITPY_WIFI_PASSWORD = "xyz"
```

Replace `abc` with your WiFi's name and `xyz` with your WiFi's password. These settings allow your device to connect to the WiFi.

Run the following code to make sure that your device is online. This test code is available in the `code/tests` folder in [13].

```
import wifi, os
wifi.radio.connect(
    ssid=os.getenv("CIRCUITPY_WIFI_SSID"),
```

```
    password=os.getenv("CIRCUITPY_WIFI_PASSWORD"))
print("my IP addr:", wifi.radio.ipv4_address)
```

The code prints a certain IP address, such as 192.168…, if everything goes well.

X.3.5 Installing an MPU6886 Driver

M5StickC PLUS2 has a 6-axis IMU (Inertial Measurement Unit), called MPU6886, which combines a 3-axis accelerometer and a 3-axis gyroscope. This project uses the accelerometer to detect the tilt of a water bottle. To enable MPU6886, install the following two libraries:

- CircuitPython Register [14]
- CircuitPython MPU6886 [15]

Visit the "releases" section on each webpage and download a library package file such as **`busdevice-9.x-mpy-...zip`**. "9.x" in the file name means that the package is compatible with CircuitPython 9.x. Unzip it and find a .mpy file (or a set of .mpy files) under the **`lib`** folder. Copy the .mpy file(s) to the **`lib`** folder in your CircuitPython file system.

Run the following code to confirm that your code can use the accelerometer. This test code is available in the **`code/tests`** folder in [13].

```
import board, mpu6886
i2c = board.I2C()
mpu = mpu6886.MPU6886(i2c)
accel = mpu.acceleration
print(accel)
```

The code prints three float numbers in a tuple, such as (0.5, 0.5, 9.8), if everything goes well. These numbers indicate the accelerometer's readings (in m/s^2) along the X, Y and Z axes, respectively. In M5StickC PLUS2, the X, Y, and Z axes are set up for the accelerometer as shown in Fig. 5. Therefore, if the device is level, the first two numbers in the triplet should be 0 (or close enough to 0) and the last number should be close to 9.8 (i.e., gravitational acceleration).

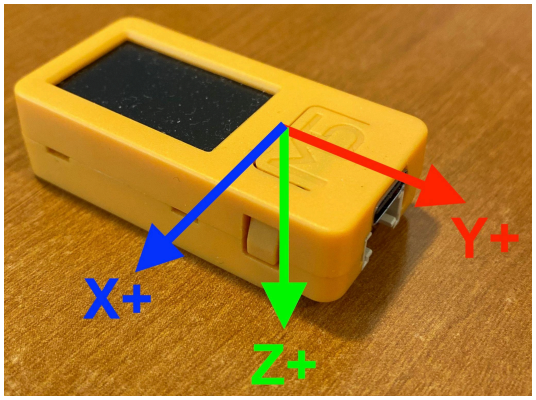


Fig. 5 Sensing Axes of an M5StickC PLUS2

X.3.6 Installing an ST7789 Driver

M5StickC PLUS2 has a display, called ST7789v2. This project uses it to show the tilt of a water bottle in degrees (Fig. 3). To enable ST7789v2, install the following two libraries:

- CircuitPython ST7789 [16]
- CircuitPython Display Text [17]

Run **`screen.py`** in the **`code/tests`** folder in [13] to test the display. If everything goes well, the code should display “Hello World” in color.

X.4 Writing and Running CircuitPython Code

Sip Log runs the following CircuitPython code, which is available as **`code.py`** in the **`code`** folder in [13]. When M5StickC PLUS2 boots, CircuitPython looks for **`code.py`** in your file system and runs it automatically. Place this code, **`display.py`** and **`kintone.py`** in the top/root folder of your CircuitPython file system. **`display.py`** and **`kintone.py`** are also available at [13].

```
import board, mpu6886, time, math
import display, kintone

SUB_DOMAIN  = ""
APP_ID      = ""
API_TOKEN   = ""

DEMO_MODE               = True
ACCEL_SENSING_INTERVAL = 1    # in seconds
ALERT_INTERVAL          = 1200 # in seconds

i2c = board.I2C()
```

```
mpu = mpu6886.MPU6886(i2c)
accelSensingCount = 0

def calcPitch(x, y, z):
    accelMagnitude = math.sqrt(x**2 + y**2 + z**2)
    yNormalized = y/accelMagnitude
    if z >= 0:
        pitchY = math.asin(-yNormalized)
    else:
        pitchY = math.acos(-yNormalized) + math.pi/2
    return pitchY

if DEMO_MODE:
    display.init()
    time.sleep(3)
else:
    display.init(demoMode=False)

while True:
    accel = mpu.acceleration
    x = accel[0]
    y = accel[1]
    z = accel[2]

    pitchRadian  = calcPitch(x, y, z)
    pitchDegrees = round( math.degrees(pitchRadian) )
    print("Pitch (degrees):", pitchDegrees)
    if DEMO_MODE:
        display.resetText("Pitch:" + str(pitchDegrees))
    accelSensingCount += 1

    if pitchDegrees >= 45:
        kintone.uploadSipLog(SUB_DOMAIN, APP_ID, API_TOKEN,
                             pitchDegrees)
        accelSensingCount = 0

    if accelSensingCount > int(ALERT_INTERVAL/ACCEL_SENSING_INTERVAL):
        alertText =f"No water intake for {int(ALERT_INTERVAL/60)} mins"
        kintone.uploadAlert(SUB_DOMAIN, APP_ID, API_TOKEN,
                            alertText)
        accelSensingCount = 0

    time.sleep(ACCEL_SENSING_INTERVAL)
```

To run this code properly, install the following two libraries:

- CircuitPython ConnectionManager [18]
- CircuitPython Requests [19]

In addition, three blue lines in the code need to be completed with a subdomain name, application ID, and API token of the Kintone cloud database. See [6] for more details about these parameter settings.

`code.py`, **`display.py`** and **`kintone.py`** have about 220 lines of code in total. Their total size is 6.1 KB. Libraries consume 34.8 KB. The total size of 40.9 KB is small enough for the 8 MB flash memory space in M5StickC PLUS2.

X.4.1 Detecting Water Intake via Tilt Sensing

`code.py` reads the accelerometer every second to calculate the tilt of the water bottle and detect a water intake. Sip Log defines the tilt as the pitch angle between the horizontal ground surface and the positive side of the accelerometer's Y-axis (Fig. 6). See Fig. 5 for the direction of the Y-axis. This project assumes that the negative side of the Y-axis faces the person using the bottle to drink water.

Fig. 6 Tilt of a Water Bottle as the Pitch Angle of the Accelerometer's Y-Axis

Once the pitch angle is calculated with the accelerometer, it is shown on the display (in degrees) (Fig. 3 and Fig. 7). It is close to zero when the bottle is level, and it increases as the bottle tilts. By default, Sip Log detects a water intake when the angle exceeds 45 degrees.

Fig. 7 Pitch Angle (in Degrees) Shown on the Display

The pitch angle is calculated with the formulae that describe the relationship between the pitch angle $\boldsymbol{\theta}$ and the y-component of the gravitational acceleration. Figs. 8 and 9 show how to calculate $\boldsymbol{\theta}$ when $\boldsymbol{\theta}$ <= 90° (i.e., g_z >=0) and $\boldsymbol{\theta}$ > 90° (i.e., g_z < 0), respectively. In both figures, the gravitational acceleration is denoted by g, and its y-component is denoted by g_y. In Fig. 8, $\boldsymbol{\theta}$ is congruent to its vertical angle and the angle of the right triangle with the leg g_y. Inside of this right triangle, a similar triangle with a hypotenuse of 1 and leg g_y' can be drawn. As a result, g_y' = sin$\boldsymbol{\theta}$. g_y' is negated in the formula because the accelerometer reports g_y as a negative value. In Fig. 9, $\boldsymbol{\theta}$ is greater than 90°. $\boldsymbol{\theta}$' is congruent to its vertical angle, which is the angle of the right triangle with the leg g_y. Inside of this right triangle, a similar triangle with hypotenuse 1 and leg g_y' can be drawn. As a result, $\boldsymbol{\theta}$' = $\cos^{-1}(-g_y')$. Since this formula uses $\boldsymbol{\theta}$', $\pi/2$ radians, or 90°, is added to obtain $\boldsymbol{\theta}$.

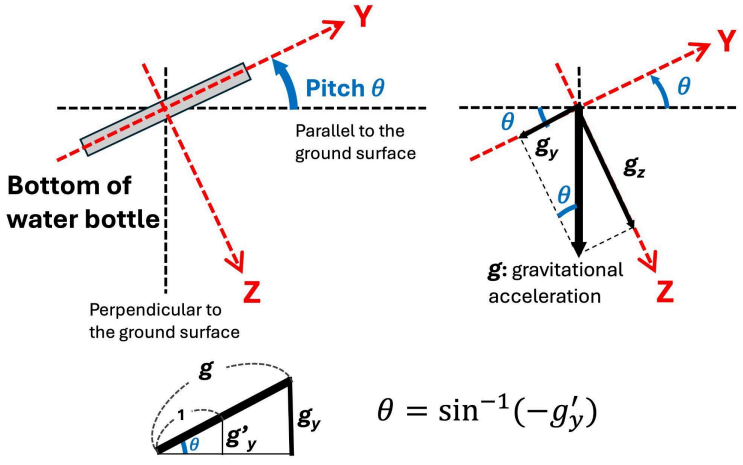
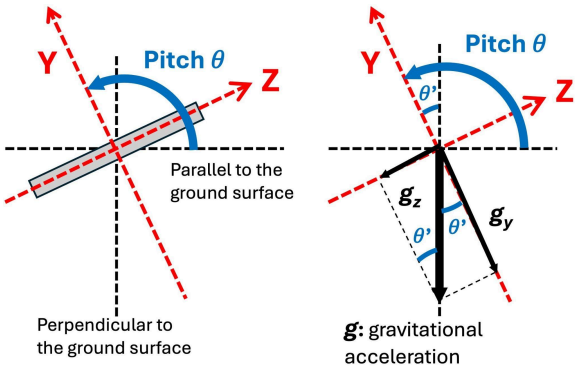


Fig. 8 Calculation of the Pitch Angle (θ) when θ <= 90°



$$\theta' = \cos^{-1}(-g'_y)$$

$$\theta = \theta' + \pi/2$$

Fig. 9 Calculation of the Pitch Angle (θ) when θ > 90°

In the CircuitPython code (**`code.py`**), the function **`calcPitch()`** takes x-, y- and z-components of the gravitational acceleration in radians, calculates the pitch angle with the formulas in Figs. 8 and 9, and returns it in radians. The returned pitch angle is converted from radians to degrees with **`math.degrees()`**.

By default, Sip Log calls **`calcPitch()`** every second. You can customize this interval by changing a constant value in **`code.py`**: **`ACCEL_SENSING_INTERVAL`**.

X.4.2 Displaying the Tilt of a Water Bottle

Sip Log continuously shows the current pitch angle in degrees on its display (Figs. 3 and 7). It calls **`resetText()`** in **`display.py`** to refresh the display. You can customize the display's colors by changing three constant values in **`display.py`**: **`BACKGROUND_COLOR`**, **`FOREGROUND_COLOR`**, and **`TEXT_COLOR`**.

X.4.3 Recording Water Intake in the Kintone Cloud Database

If the current pitch angle exceeds 45 degrees, Sip Log detects a water intake and logs it in Kintone. It uploads the current time stamp and pitch angle with the function **`uploadSipLog()`** in **`kintone.py`**. The user can browse water intake history with a Web browser or Kintone's mobile apps [20, 21]. Fig. 10 shows an example intake history and its hourly summary as a bar chart. This chart visualizes a trend of water intakes and helps estimate the amount of water consumption.

Fig. 10 Water Intake History (Left) and its Hourly Summary (Right) in Kintone

X.4.4 Sending Hydration Reminders

If no intakes are logged for 20 minutes, Sip Log records a hydration alert in Kintone by calling the function **`uploadAlert()`** in **`kintone.py`**. You can customize this 20-minute threshold by changing the **`ALERT_INTERVAL`** constant in **`code.py`**. Fig. 11 shows an example hydration alert recorded in Kintone.

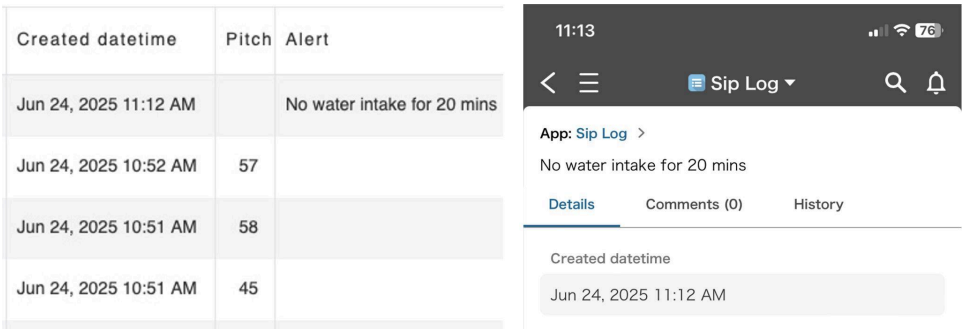


Fig. 11 Hydration Alert in Kintone

Kintone is configured to issue a notification to its mobile application when a hydration alert is recorded. Fig. 12 shows an example iPhone notification issued by Kintone. This works as a reminder to drink water.

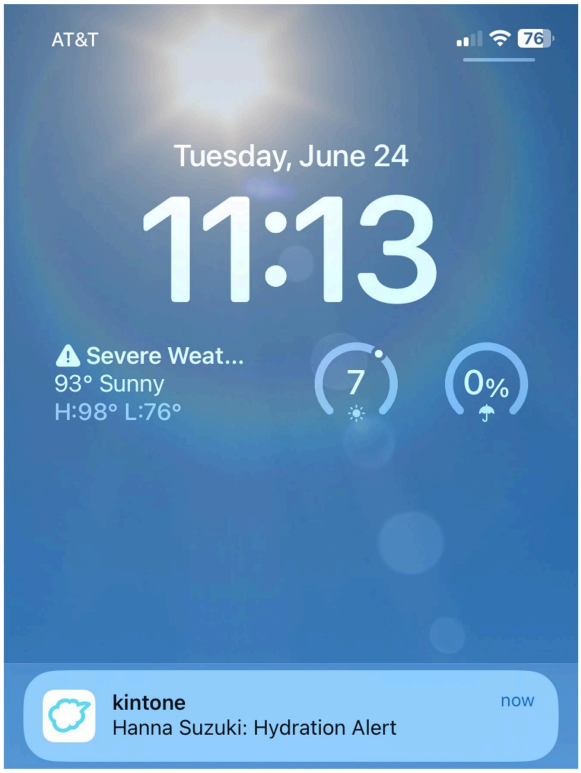


Fig. 12 Hydration Reminder from Kintone

X.4.5 Adjusting Battery Life

With the battery fully charged (200 mAh), Sip Log can operate for 2 to 2.5 hours if its display remains on. This is reasonably long enough for a sports practice/event.

You can turn off the display by setting `False` to the `DEMO_MODE` constant in `code.py`. (`True` is set to the constant by default.) Sip Log can extend the battery life by 30 minutes if the display is turned off.

X.5 Future Work

Sip Log still has room to improve. A potential improvement is to use two axes of the accelerometer, Y- and X-axes, to detect water intake. Currently, Sip Log uses the Y-axis only (Fig. 5). This requires a specific way to hold a water bottle and take in water, as the negative side of the Y-axis faces the user. This requirement can be eliminated by considering both Y- and X-axes to calculate the tilt of a water bottle.

Another improvement is to automatically calculate the amount of water consumption and show it as an hourly bar chart in Kintone. This can be done by multiplying the count of intakes by the

average amount of water consumption per intake (e.g., 1.25 to 1.3 oz per intake). This can be implemented with a “calculated” field in Kintone. It is a special database field that automatically calculates a value with an embedded equation and other field values.

Moreover, Sip Log could automatically adjust the interval to issue hydration reminders. Currently, it is fixed to 20 minutes. If the weather is quite hot and strenuous, Sip Log could shorten the interval to remind the user of hydration more often. Similarly, if the weather conditions are relatively mild or if the user is inside in a controlled environment, Sip Log could extend the interval to send out reminders less often.

X.6 Conclusion

Sip Log computerizes a water bottle with a battery-operated ESP32 microcontroller to keep track of water intakes, record intake logs in the cloud, help estimate water consumption, and issue hydration reminders via smartphone notifications. Built as a boot attachment for a bottle, it is light as 3.5 oz (96 grams) with the battery life of 2.5 to 3 hours, which is feasible enough for a sports practice/event. Sip Log is designed to help athletes practice healthy hydration habits.

References

[1] Johns Hopkins Medicine, Sports and Hydration for Athletes: Q&A with a Dietitian. https://www.hopkinsmedicine.org/health/wellness-and-prevention/nutrition-and-fitness/sports-and-hydration-for-athletes

[2] B. McDermott, P. Brendon P, L. Armstrong, et al., National Athletic Trainers’ Association Position Statement: Fluid Replacement for the Physically Active. Journal of Athletic Training, 52(9):877–895, 2017.

[3] N. Shaheen, A. Alqahtani, H. Assiri, et al., Public Knowledge of Dehydration and Fluid Intake Practices: Variation by Participants’ Characteristics. BMC Public Health, 18(1):1346, 2018.

[4] USTA Sport Science: Heat and Hydration Concerns for Tennis Players, July 2016.

[5] Tara Gidus Collingwood and Trish Kellogg, Hydration Matters, USTA, January 2017.

[6] https://kintone.dev/

[7] National Federation of State High School Associations, Position Statement and Recommendations for Maintaining Hydration to Optimize Performance and Minimize the Risk for Exertional Heat Illness, 2018.

[8] https://docs.m5stack.com/en/core/M5StickC%20PLUS2

[9] https://learn.adafruit.com/how-to-install-drivers-for-wch-usb-to-serial-chips-ch9102f-ch9102/

[10] https://circuitpython.org/

[11] https://circuitpython.org/board/m5stack_stick_c_plus2/

[12] https://thonny.org/

[13] https://github.com/HSSBoston/sip-log

[14] https://github.com/adafruit/Adafruit_CircuitPython_Register

[15] https://github.com/tkomde/CircuitPython_MPU6886

[16] https://github.com/adafruit/Adafruit_CircuitPython_ST7789

[17] https://github.com/adafruit/Adafruit_CircuitPython_Display_Text

[18] https://github.com/adafruit/Adafruit_CircuitPython_ConnectionManager/

[19] https://github.com/adafruit/Adafruit_CircuitPython_Requests/

[20] https://apps.apple.com/us/app/kintone/id674312865

[21] https://play.google.com/store/apps/details?id=com.cybozu.kintone.mobile