Co-funded by the
Erasmus+ Programme
of the European Union

Kozienice

Zespol Szkol Nr 1 im. Legionow Polskich

E N E R G Y A N D E N V I R O N M E N T

12.10. – 15.10.2021

Topic	Age	Country	Date
Drive a rover with Python Regular polygon	>15	France	Oct 2019

Drive a programmed rover in Python

Experimental setup materials :

1 calcultor TI

1 rover TI

1 hub (microcontroller)

Experimental setup and procedure :

Start the **TI-Innovator Rover. Choose prgm** on the calculator.

Run this program :

ÉDITEUR : PLYGONE
LIGNE DU SCRIPT 0009

```
import ti_rover as rv
n=int(input("number sides "))
d=input("lenght ")
def polyg(n,d):
  for i in range(n):
    rv.forward(d)
    rv.left(360/n)
polyg(n,d)
```

Experiment evaluation:

1) Describe and explain the passage the rover takes. You can use a marker in the hole so that the rover can trace his passage.

2) Create a new program, using a loop and a function Python.

Topic	Age	Country	Date
Drive a rover with Python Emergency stop	>15	France	Oct 2019

Drive a programmed rover in Python

Experimental setup materials :

1 calcultor TI

1 rover TI

1 hub (microcontroller)

Experimental setup and procedure :

Start the **TI-Innovator Rover. Choose prgm** on the calculator.

Run this program :

```
ÉDITEUR : LUMIERE
LIGNE DU SCRIPT 0005
import ti_rover as rv
d=rv.ranger_measurement()
print(d)
if d>0.2:
  rv.forward(10*d-1)_
```

Experiment evaluation:

1) Describe and explain the passage the rover takes.

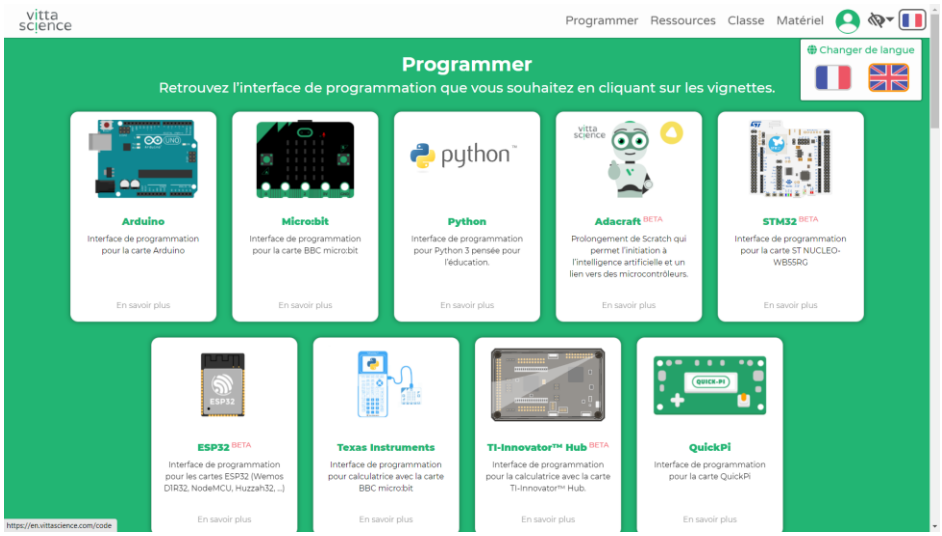
2) Create a program, using a loop that the rover travels a regular polygon.

Topic	Age	Country	Date
Programming an online temperature sensor	>15	France	Oct 2019

Simulate a Python sensor program

Experimental setup materials :

1 laptop or a computer with internet connection



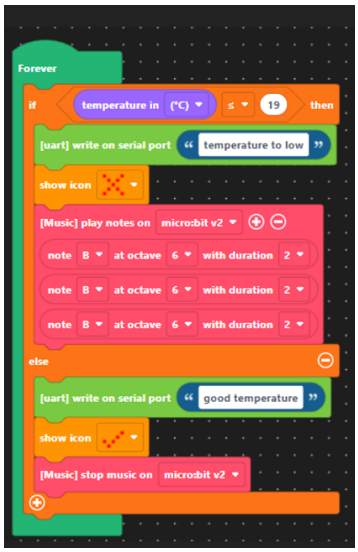
Experimental setup and procedure :

Go to the website Vittascience en.vittascience.com/python

Experiment evaluation:

1) Reproduce this program and run it. Observe the Python code.

2) Create another program of your choice, using another sensor.



Topic	Age	Country	Date
Programming a microcontroller ESP8266 for measures of temperature and light	>15	France	Oct 2021

Python sensor program

Experimental setup materials :

1 laptop or a computer with internet connection
1 microcontroller ESP8266
1 numeric sensor of temperature
1 analogic sensor of light

Experimental setup and procedure :

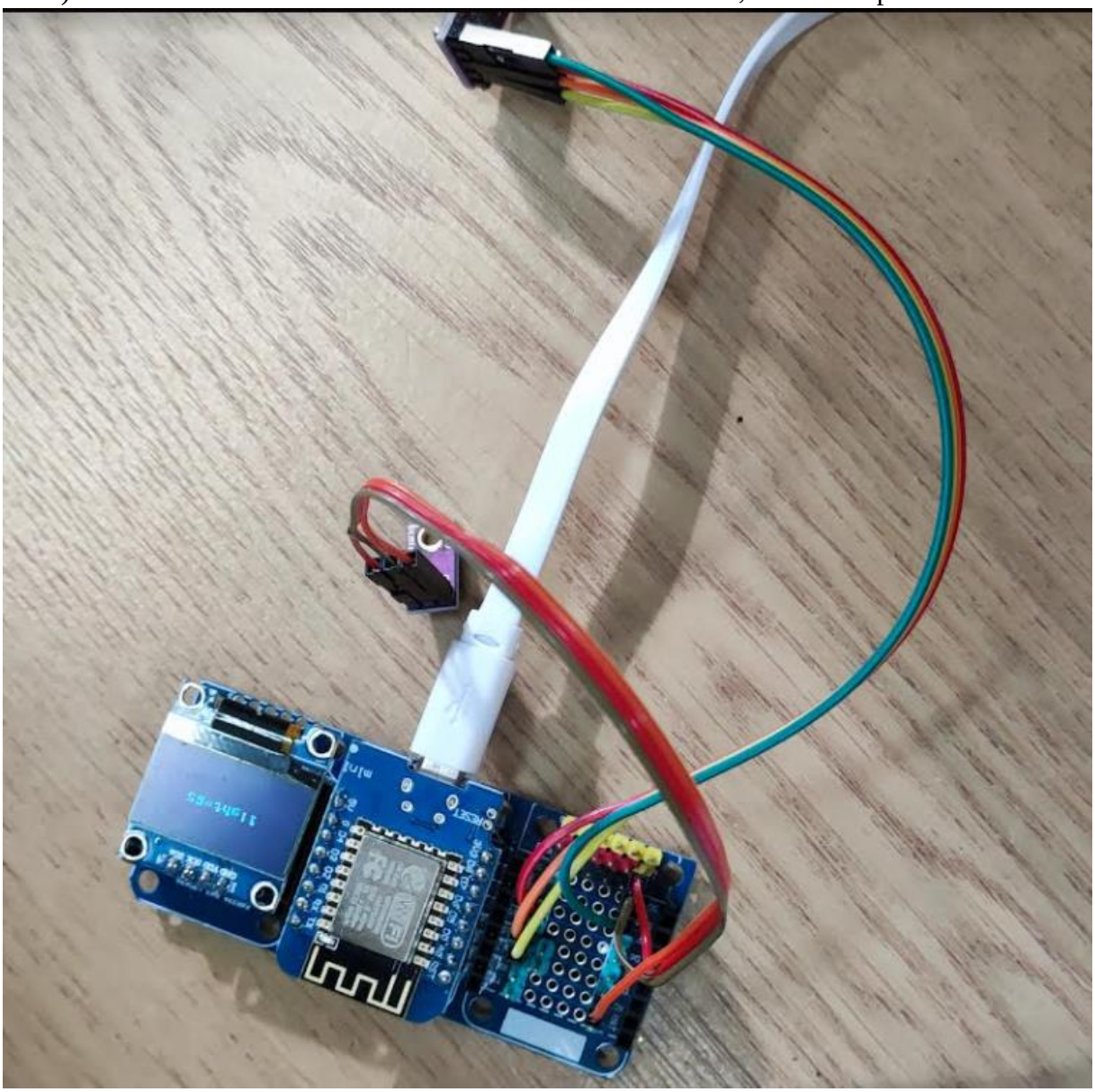
With Thonny, use the programm : *copy and paste the Python code*

```
from machine import Pin, I2C, ADC
import ssd1306
import time
i2c = I2C(-1, Pin(5), Pin(4))  # liaison serie : Pin4 = Serial Data, Pin5 = Serial Clock
display = ssd1306.SSD1306_I2C(128, 64, i2c)  # resolution de 128x64
buffer = bytearray(2)

CaptLum = ADC(0)

while True:
    M = CaptLum.read()
    buffer = i2c.readfrom(0x49, 2)
    N = (buffer[0] << 3) | (buffer[1] >> 5)
    Temp = N * 0.125  #  resolution LM75A is 0,125 °C
    display.fill(0)  # Efface l'écran
    display.text("Temp= " + str(Temp) + " C", 0, 20)# + concatenation operator
    #function str() convertes number on string
    if M >= 1023 : M = 1023
    display.text("light = " + str(M), 0, 10)
```

```
display.show()
time.sleep_ms(1000)  # each one seconde
```

Experiment evaluation:

1) Connect the two sensor on the microcontroller ESP8266, like of the photo :



2) Run this program and look at the results displayed on the OLED screen.

Topic	Age	Country	Date
Centripetal acceleration (Phyphox)	>14	Germany	Oct 2021

- Rotating “machine” (salad spinner, special chair, wheel etc.)

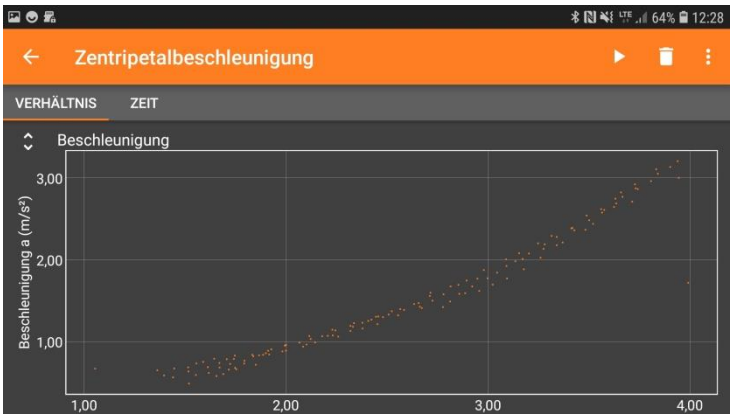
This program visualizes centripetal acceleration as a function of angular velocity.

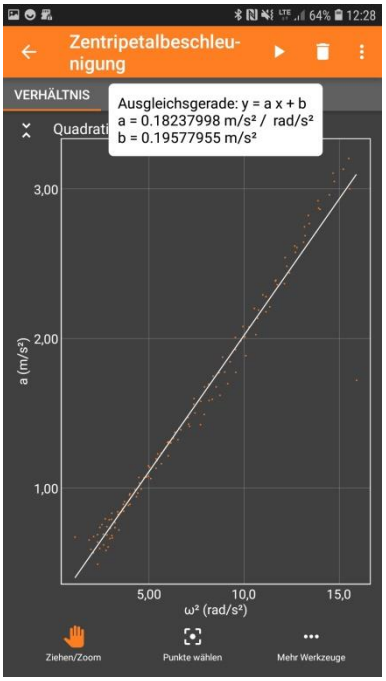
To see this, you need to measure at different angular velocities but a fixed radius.

Smartphone fixed on a rotating wheel

Smartphone fixed inside a salad spinner:

After the experiment you can analyse the graphs in Phyphox by tapping on a diagram:





The relation $a_z \propto \omega^2$ is a logical consequence from both diagrams.

Topic	Age	Country	Date
Distance measurement with phyESPx	>14	Germany	Oct 2021

Rolling bottle

Experimental setup materials:

1 phyESPx server with attached HC-SR04 ultrasonic measurement hat

1 tablet/laptop for evaluation

1 bottle (filled with water completely)

Experimental setup and procedure:

Turn on the **phyEXPx** server and access the web interface by scanning the QR code on top of the server or by entering the IP address (you will find it right beneath the QR code) in a browser tab. Put the sensor kit down on a flat surface (eg. a table) and make sure the distance sensor is **facing parallel to the surface**. **Remove all obstacles in the range of the sensor** (about 30° - 1,5 meters) and place the bottle right in front of it.

Now you can start the measurement by pressing the **start** button (or the **play** button on the remote device). If you can see the first data values displayed, you can **roll the bottle away from the sensor** by giving it **a gentle push**.

Once the bottle has stopped rolling or is out of sensor range, you can stop the measurement and evaluate your data.

Experiment evaluation:

1) Describe the t-x-diagram.

2) Try to imagine the fitting t-a-diagram.

3) Guess why the distance might only be accurate at 20°C

Topic	Age	Country	Date
Wireless weather sensor	>14	Italy	Oct 2021

The Wireless Weather Sensor is an all-in-one instrument for monitoring complex environmental conditions. It houses several sensing elements within a single unit to provide 19 different measurements.

We use the sensor as a handheld instrument to study microclimates and record ambient conditions relevant to environmental phenomena, such as weather and light measurements:

- Ambient Temperature
- Barometric Pressure
- Relative Humidity
- Wind Speed
- Ambient Light (lux)

We transmit data wirelessly to our device (computer, tablet and mobile) for classroom analysis when group activities are constrained by time, within SPARKvue software.

Procedure

1. Select a location for your experiment according to your teacher's instructions.
2. Start a new experiment on the data collection system (Sparkwue)
3. Attach the weather sensor to the data collection system.
4. Display barometric pressure, temperature, relative humidity and wind speed in a table.
5. Change the sample rate to once every 10 minutes.
6. Keep the data collection system off the ground.
7. Keep the direct sun from shining on your system by placing a weather shield over it.
8. Start data recording.
9. After the data has been collected for a few minutes, stop recording data.
12 Save your experiment.
13 Return the equipment and the data collection system to the classroom.

Topic: Energy Field experiments. Data logger	Age >15	Country Poland	Date October 2021

Function, realisation

In this manual, we will show you how to record measurement results during field experiments. We will describe an easier way - saving the results on the internal Pico memory. We will use the internal thermometer of the Raspberry Pi Pico as the sensor. This script can be easily adapted to record measurements from other sensors, e.g. smog, pressure, water purity or many others, including the time of measurement or coordinates describing the place of measurement taken from a GPS connected to Pico. We will show you how to save the measurement results in the Pico's internal memory as a CSV file, which can be easily processed later by a spreadsheet or other scripts using the **Matplotlib** library for plotting graphs. We will also present a more difficult solution - saving the results on a MicroSD card.

Hardware required

- Microcontroller Raspberry Pi Pico with Micropython 1.18
- Cytron Maker Pico docking station for Pico
- Optional 3V3 LCD 1602 I2C display from Seeedstudio with Grove Socket,

Materials required

- micro usb 2.0 high speed (15cm or 30 cm),
- powerbank,

- MicroSD card (max 16 GB),
- optional 1 standard Grove wire.

Software required

- Thonny (thonny.org)
- library for LCD (script with LCD only): https://files.seeedstudio.com/wiki/Grove_Shield_for_Pi_Pico_V1.0/Libraries.rar
- sdcard library (script with MicroSD reader support only): https://github.com/micropython/micropython/blob/master/drivers/sdcard/sdcard.py

Setup software

Below Python code for saving in microcontroller using Thonny App. Write the script in Thonny, then select "File/Save" and select Raspberry Pi Pico as destination. Remember please to add .py at the end of filename.

```
from utime import sleep
from machine import ADC
#thermometer initialization (on the Pico board)
sensor_temp = ADC(ADC.CORE_TEMP)
#Factor for converting ADC indications to voltage
conversion_factor = 3.3 / (65535)
#filename assignment and opening for writing
file = open("measurements.csv", "w")
while True:
    reading = sensor_temp.read_u16() * conversion_factor
    #Temperature calculations based on the sensor characteristics
    #rounding to one decimal place
    temperature = round(27 - (reading - 0.706)/0.001721,1)
    print(temperature)
    file.write(str(temperature) + "\r\n")
    #flush()-works like the "Close" statement, but the file remains open
    file.flush()
    sleep(5)
```

Code analysis

The **machine** module contains specific functions related to the hardware on a particular board. We import methods related to ADC- Analog Digital Converter (line 2).

The **utime** module (line 1) provides time functions. In this case we use **sleep** method for delay.

In lines number 4,6 we make thermometer initialisation (on the Pico board) and set factor for converting ADC indications to voltage.

You can also initialise another sensor here. In the eighth line we make filename assignment and opening for writing Any data obtained from the sensor are saved in the file: **measurements.csv**. The file is opened for writing (w), its previous contents are erased.

On lines 9-18 there is an infinite loop.

Lines 10 and 13 are used for read the data from the sensor and convert them using the appropriate conversion factor. The "**round**" method rounds the result to one decimal place

Lines 15 and 17 are essential for writing to the file.

The flush() method in Python file handling clears the internal buffer of the file. In Python, files are automatically flushed while closing them. However, a programmer can flush a file before closing it by using the flush() method and finally save results in a file.

If we want to save the values of several variables in a file, it is worth using tabs to separate columns:

file.write(str(press)+"\t"+str(temp)+"\t"+ str(humid)+"\r\n")

In above example we write values of 3 variables: press. temp and humid as strings separated by tabs.

Notice: \t means tab, \r means return, \n means "new line".

Stored file has the the name with CSV extension, then this text file may be interpreted as database of results by many apps, easy to analyse and process.

Data recording in time

The last line "**sleep(5)**" makes the temperature readable every four seconds.

If you want to measure the value every minute, you have to change 5 to 60, one per hour: 3600. This value can be freely changed, but not more often than the maximum fixed for a given sensor. In practice, there is no need to write frequently, because the temperature, for example, does not change quickly. If we do not save the results often, Pico memory will be enough for a longer time

Optional version with lcd screen support

```
from utime import sleep
from machine import ADC, Pin, I2C
from lcd1602 import LCD1602
#thermometer initialisation (on the Pico board)
sensor_temp = ADC(ADC.CORE_TEMP)
#Factor for converting ADC indications to voltage
conversion_factor = 3.3 / (65535)
#initialisation: I2C bus for LCD  connected to GP6, GP7
i2c = I2C(1,scl=Pin(7), sda=Pin(6), freq=400000)
#LCD initialisation
d = LCD1602(i2c, 2, 16)
d.display()
#Clear screen
d.clear()
#filename assignment and opening for writing
file = open("measurements.csv", "w")
while True:
    reading = sensor_temp.read_u16() * conversion_factor
    #Temperature calculations based on the sensor characteristics
    #rounding to one decimal place
    temperature = round(27 - (reading - 0.706)/0.001721,1)
    print(temperature)
    d.setCursor(0, 0)
    d.print("int. temperature:")
    d.setCursor(4, 1)
    d.print(str(temperature)+"C")
    file.write(str(temperature) + "\r\n")
    #flush()-works like the "Close" statement, but the file remains open
    file.flush()
    sleep(5)
```

We must add libraries required to handle the display. In above script lines 9-14 define the display and lines 23-26 are for printing results on LCD display. **Note that according to the code, the lcd should connect to GP6, GP7.**

Sample content of the file with measurement results:

```
16.3
19.6
19.6
19.6
19.6
19.6
19.1
19.6
19.1
20.0
19.6
19.1
19.6
19.6
```

Performance of the experiment

If you want have autonomous system then save the script on the Pico with special name: **main.py.** This script will start automatically (No need to connect to a computer). In this case you can power the Pico from powerbank. Firstly you have to connect board to powerbank (or other type of electric source) through the micro USB wire.

Warning! If you use the name **main.py** (with autostart) and you want save the file with results, detach the LCD screen. This will allow you to terminate the script before erasing the file.

Evaluation, testing:

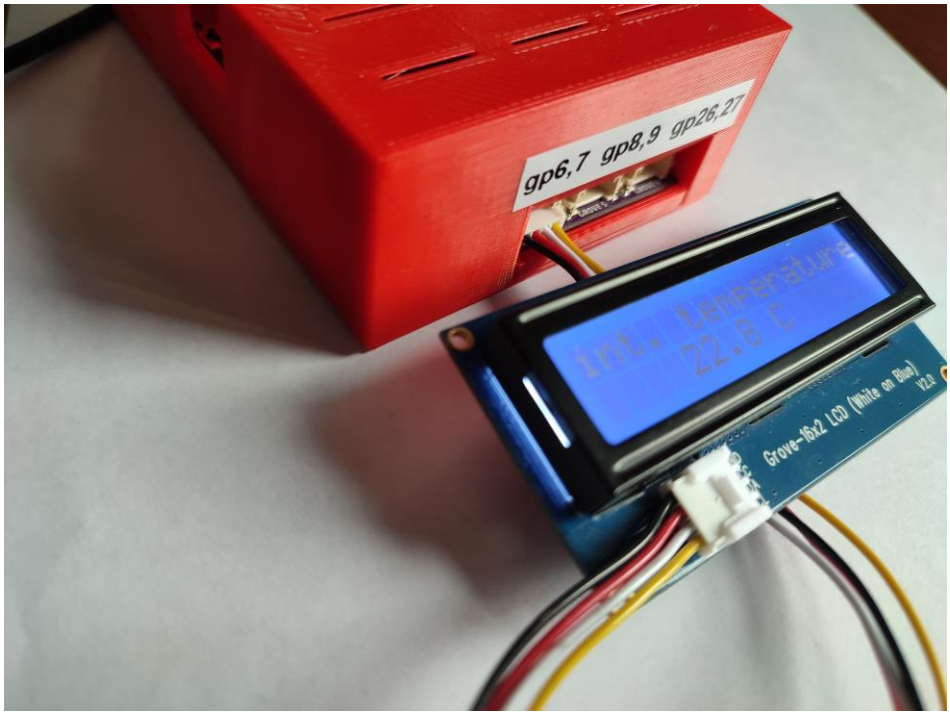
- you must remember, that every time you start the script, then you erase data in file
- you can adjust information stored in file or use with different sensor
- estimate the file size assuming that the script runs for 1 day and the temperature is recorded once every 60 seconds
- open the CSV file with a spreadsheet and use it to chart (from Pico internal memory or MicroSD card.
- consider what sensors you could use for field measurements

```
from utime import sleep
from machine import Pin, ADC, SPI
import sdcard
import uos
""" SPI mode Cytron Maker Pico shield
micro SD socket (description from the board)
GP10-SCK (SCLK)
GP11-SDI (SDI<---mosi)
GP12-SDO (SDO<---miso)
GP15-CSN (chip select)
All internal connections, no wires needed,
you don't have to connect anything"""
# Assign chip select (CS) pin (and start it high)
cs = Pin(15, Pin.OUT)
# Intialize SPI peripheral (start with 1 MHz)
spi = SPI(1,
                  baudrate=1000000,
                  polarity=0,
                  phase=0,
                  bits=8,
                  firstbit=SPI.MSB,
                  sck=Pin(10),
                  mosi=Pin(11),
                  miso=Pin(12))
# Initialize SD card
sd = sdcard.SDCard(spi, cs)
# Mount filesystem
vfs = uos.VfsFat(sd)
uos.mount(vfs, "/sd")
#filename assignment and opening for writing
file = open("/sd/measurements.csv", "w")
#termometer initialisation
sensor_temp = ADC(ADC.CORE_TEMP)
conversion_factor = 3.3 / (65535)
while True:
    reading = sensor_temp.read_u16() * conversion_factor
    temperature = round(27 - (reading - 0.706)/0.001721,1)
    print(temperature)
    file.write(str(temperature) + "\r\n")
    #flush()-works like the "Close" statement, but the file remains open
    file.flush()
    sleep(5)
```

Go further**:

- write a Python script that will automatically display the chart based on the CSV file using the **Matplotlib** library
- prepare a project that reads the coordinates from the GPS module, adjust it to work with the logger.
- write a Python script that will read the **measurements.csv** file from the MicroSD card and display its contents in the console

Below data logger inside Pico with our 3D printed housing



Topic: Energy	Age	Country	Date
DS18B20-energy efficiency measurement.	>14	Poland	October 2021

Function, realisation

Construction of a measuring instrument based on microcontroller with Micropython and digital waterproof thermometer, which can be used for measurement of energy efficiency, equipment efficiency.

Hardware required

- Microcontroller Raspberry Pi Pico with Micropython 1.18 or above
- Cytron Maker Pico docking station for Pico
- DS18B20 waterproof probe -digital thermometer (3.3V for Pico) with **onewire** support
- 3V3 LCD 1602 I2C display from Seeedstudio with Grove socket,
- PC or Mac computer.

Materials required

- minibreadboard
- 4,7 kOhm resistor
- 1xstandard Grove wire, 1xGrove male wire
- ARK4 screw terminal for breadboard (minimum screw 3 inputs)
- micro usb 2.0 high speed cable(15cm or 30 cm)
- LCD library: https://files.seeedstudio.com/wiki/Grove_Shield_for_Pi_Pico_V1.0/Libraries.rar
- powerbank (for standalone version)
- paper for note

Software required

- ThonnyApp (thonny.org),
- ds18b20 use library included in MicroPython (onewire, ds18x20),
- library for LCD.

Using Thonny App You can save library inside lib folder in Pico memory.

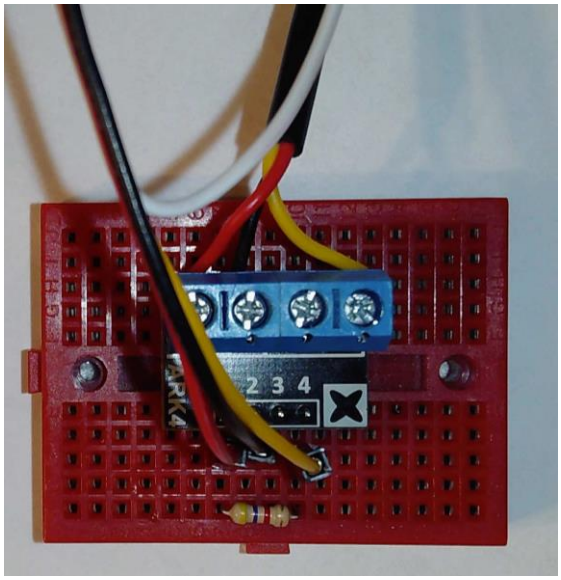
Technical specifications of DS18B20 digital thermometer:

- Supply voltage: 3 V to 5 V

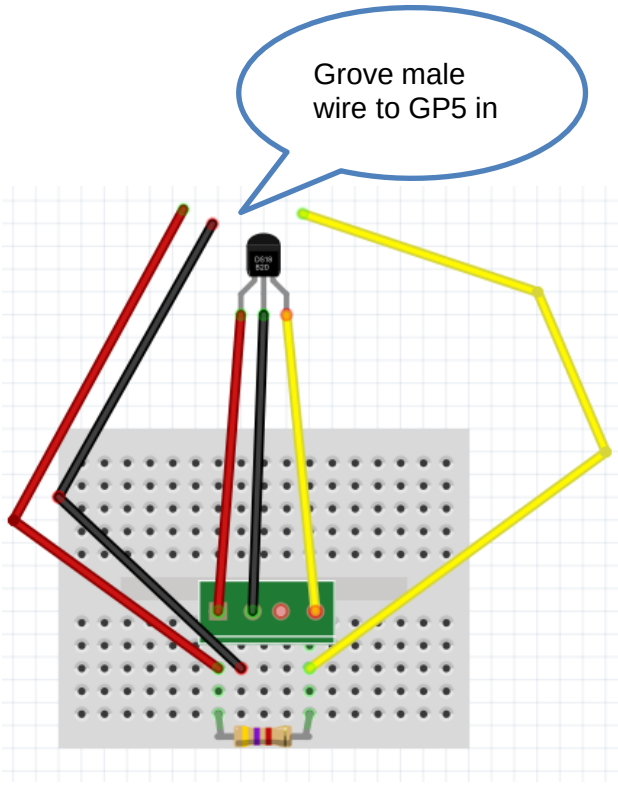
- Operating temperature range: -55°C to 125°C
- Storage temperature: from -55°C to 125°C
- Accuracy: ±0.5°C in temperature range from -10°C to 85°C

Setup Hardware

DS18B20 digital thermometer support 1 wire. This allows multiple thermometers to be connected to one signal line. They are recognised by a unique address in each thermometer. This is very convenient if you need multiple thermometers. We don't even need to modify the program, it will work fine.



DS18B20 thermometer pins	Raspberry Pi Pico pins	Grove male wire
VCC (red)	3V3	red
GND (black)	GND	black
Signal (yellow)	GP5	yellow
not connected	GP4	white



Important! Use mini breadboard and a 4.7k Ohm resistor as a pull-up resistor to the signal pin of the thermometer. Best of all, the circuit will not work.

Resistor connection:

Resistor 4.7kOhm	Grove male wire	DS18B20 ARK 4 screw terminal
1. leg line	red	VCC (1)
---------	black	GND (2)
---------	white	not connected (3)
2. leg line	yellow	signal (4)

First resistor leg at the same line like red wire, second resistor leg at the same line like yellow - signal wire.

Note the offset of the terminal pins relative to the screw connections!

The Pico use 3.3V power (not 5!!!). We usually connect the red wire to pin 3V3 and black wire to the ground (GND). SDA and SCL are the pins used for I2C communication.
We suggest using the LCD screen to display the results. In this case, we can also build an autonomous, mobile system that can work without a computer. The use of grove cables makes the connection as easy as possible.

LCD 1602 pins	Raspberry Pi Pico pins	Grove standard wire
GND	GND	black
VCC	3V3	red
SDA	GP6	white
SCL	GP7	yellow

Raspberry Pi Pico in our 3D printed housing with Grove LCD and BME280 pressure sensor

Setup software

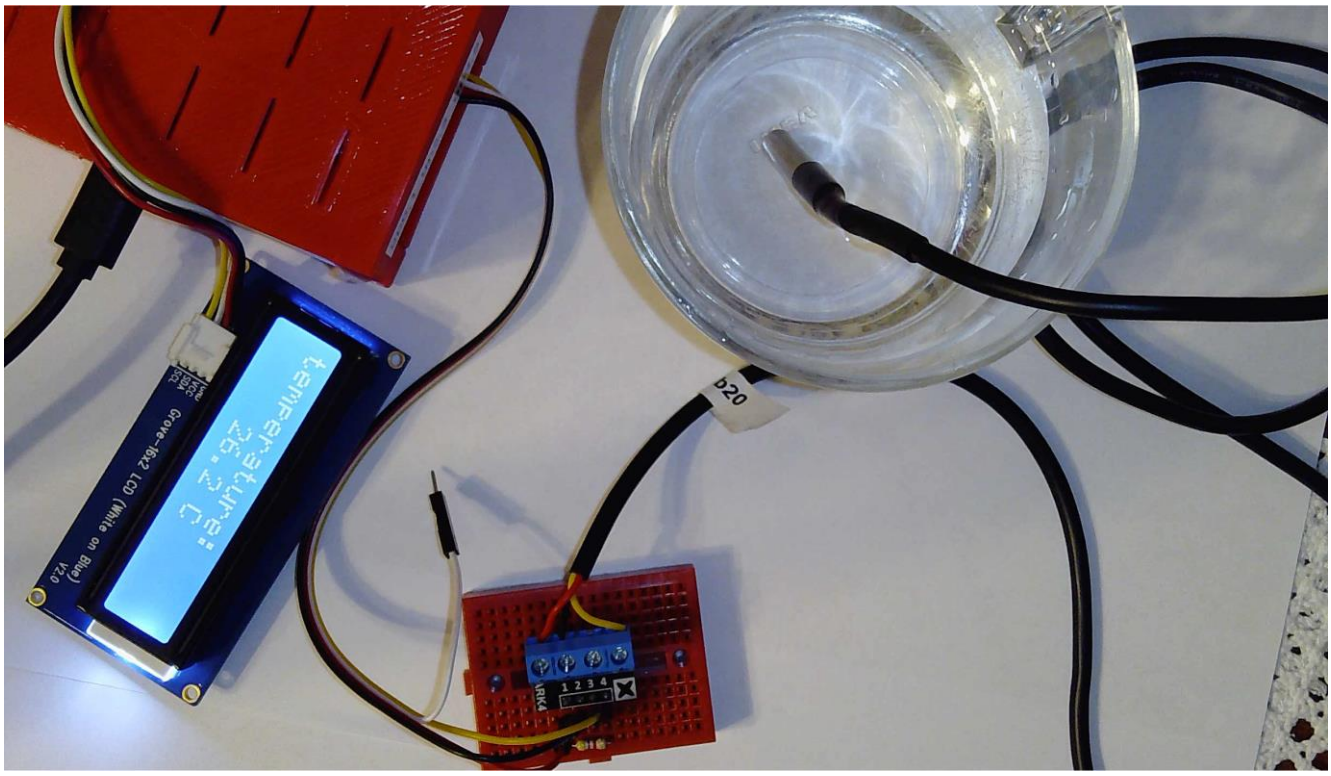
Below Python code for saving in microcontroller using Thonny App. Write the script in Thonny, then select "File/Save" and select Raspberry Pi Pico as destination. Remember please to add .py at the end of filename. Our proposal for filename: **ds18b20termdigLCD.py**

```
from machine import Pin, I2C
import onewire, ds18x20
from utime import sleep, sleep_ms
from lcd1602 import LCD1602

#signal pin connected to GP5
ds_pin = Pin(5,Pin.PULL_UP)

ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin))
roms = ds_sensor.scan()
print('Found DS devices: ', roms)
#I2C Bus initialisation
i2cbus = I2C(1,scl=Pin(7), sda=Pin(6), freq=400000)
#LCD initialisation
d = LCD1602(i2cbus, 2, 16)
d.display()
d.setCursor(0,0)
d.print("Initialisation")

while True:
    ds_sensor.convert_temp()
    sleep_ms(750)
    for rom in roms:
        d.clear()
        d.setCursor(0,0)
        d.print("temperature:")
        #print rom temperature
        print(round(ds_sensor.read_temp(rom),1))
        d.setCursor(5,1)
        d.print(str(round(ds_sensor.read_temp(rom),1))+" C")
        sleep(5)
```

Below You can find simple version of the code, which can be used with Thonny Plotter.Will be good to start with this simple version to check if everything works well.

```
from machine import Pin
import onewire, ds18x20
from utime import sleep, sleep_ms
#signal pin connected to GP5
ds_pin = Pin(5,Pin.PULL_UP)

ds_sensor = ds18x20.DS18X20(onewire.OneWire(ds_pin))
roms = ds_sensor.scan()
print('Found DS devices: ', roms)
while True:
    ds_sensor.convert_temp()
    sleep_ms(750)
    for rom in roms:
        #print rom temperature
        print(round(ds_sensor.read_temp(rom),1))
        sleep(5)
```

If you want have autonomous system then save the script on the Pico with special name: **main.py.** This script will start automatically (No need to connect to a computer). In this case you can power the Pico from powerbank.



Experiment Evaluation

After checking the correct operation of the constructed system, you can proceed to the next tests.Testing electric kettles. **Perform the experiment under the supervision of the teacher. You have to be sure that it is safe to use your thermometer kettle.**

Write down your results:

parameters	pressure	temperature	humidity
initial temperature	.....	initial temperature	.....
final temparature	100 C	final temparature	100 C
temperature difference	.....	temperature difference	.....

time to boil	.....	time to boil	.....
mass of water in the kettle	500 g	mass of water in the kettle	1000 g
water boiling time	.....	water boiling time	.....
kettle power (W)	.....	kettle power (W)	.....
efficiency (%)	.....	efficiency (%)	.....

Calculation of the heat energy needed to boil water based on the temperature and weight of the water (use the specific heat of water)

. . . . . .

. . . . . .

Calculating the real energy needed to boil water based on the boiling time and electric power of the kettle

. . . . . .

. . . . . .

Go further

This sensor can be used in many situations, where waterproof is required.

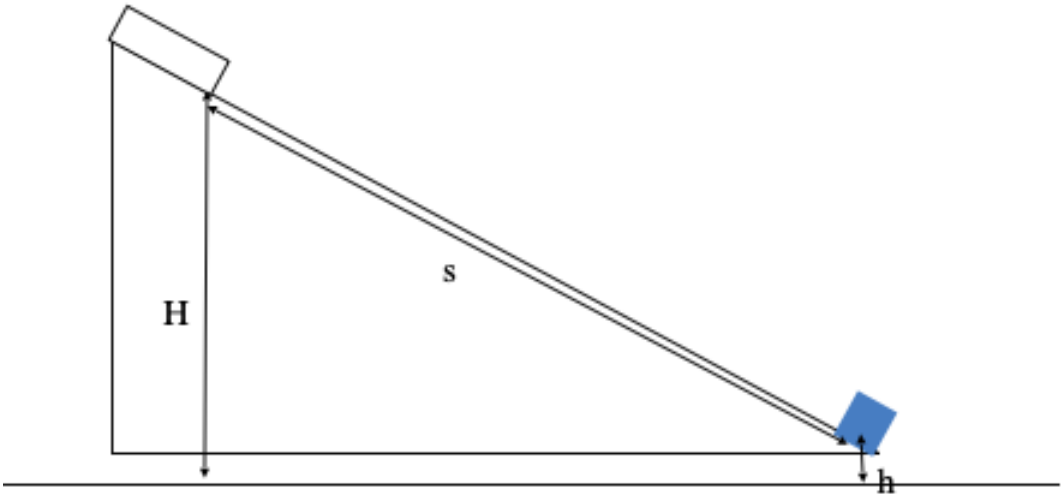
- Measure temperature air, ice water, tea
- Test the rate of changes in the temperature of the liquid, apply thermal insulation and observe the difference in cooling time
- Use a glass of 0.25 liter water placed outside the house to record changes in temperature throughout the day. You can calculate the energy changes between morning (7:00 am) and 2:00 pm. You can use our datalogger example for this
- Measure the temperature of the tea in the glass. Verify the result using a thermal camera
- The use of the presented measuring system can be found in the experiment prepared by the Portuguese school: "Joule experiment"

Topic: Energy	Age	Country	Date
Energy changes on an inclined plane	>14	Poland	October 2021

Function, realisation

Observations of energy changes during the block's movement on an inclined plane

Follow the instructions as described below



1. Place the inclined plane at an angle to the horizontal (as in the picture above)
2. place the photo gate at the end of the slope
3. Make measurements of the height of H, h and the length of S and write them down in the attached table
4. Activate the speed measurement on photo gates and release the

cuboid from the top, equal to the height H

5. Record the speed measurement results in table 1

6. Perform the action in point 4 four more times for the same height and table the results

7. Calculate the average speed from the obtained average speed Vavg

8. Using the appropriate formulas and dependencies, calculate the potential energy Ep at the height H and the kinetic energy Ek at the bottom of the inclined plane.

	mass	H	h	s		
V_1						
V_2						
V_3	V_s					
V_4						
V_5						

Calculate the value of the potential energy at the top of the slope:

$E_p=mg(H-h)$

Take the value $g = 9{,}81\ m/s^2$

Calculate the value of the kinetic energy of the block at the bottom of the inclined plane:

$$E_k=\frac{mV^2}{2}$$

Find the value of the friction force T.

Ideally, when friction does not exist, the law of conservation of mechanical energy in motion can be applied. This means that the total energy Et, i.e. the sum of the kinetic energy Ek and potential Ep, does not change during the movement. During the movement of the block, the potential energy Ep turns into kinetic energy Ek.

At the time of the start, the total energy is equal to the potential energy $E_t = E_p = mgh$,

At the end of the tier, all potential energy is converted into kinetic energy:

$E_t = E_k = 0.5\ mV^2$

Ideally, we can write:

$mgh = 0.5\ mV^2$

In the case of a frictional motion, the frictional force does the work and the energy is lost, therefore the factor $W = Ts$ must appear in the energy balance, where s - the path on which the constant friction force T acts.

At the top of the inclined plane we have: $E_t = E_p = mgh$

However, at the end of the equation $W + E_k = Ts + 0.5mV^2$.

Therefore, in this case, the energy balance has the form: $Ts + 0.5mV^2 = mgh$.

Hardware required

- 2x Vernier Photogates

- Labquest mini interface (microcontroller)
- (instead of above you can use 2 photo gates with IR sensor and Pico microcontroller with MicroPython)
- Pc or Mac

Materials required

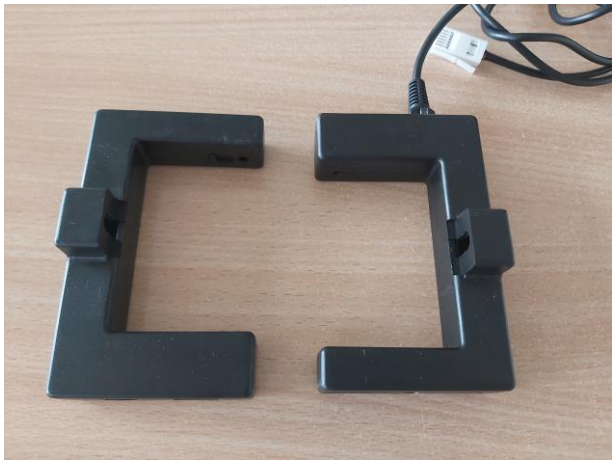
- inclined plane
- block
- cart

Software required

- Vernier Logger Pro or Graphical Analysis (or Thony App if you use Pico)

Experiment Evaluation

System testing:




PRACA DYPLOMOWA

Go further

-determination of the friction coefficient

-performing an experiment with a different slope angle

Topic: Energy Energy changes in the harmonic motion of mass on a spring (determination of the spring constant)	Age >14	Country Poland	Date October 2021

Function, realisation

To calculate the potential spring energy, it is necessary to measure the spring constant k. Hooke's law states that the force of the spring is proportional to its elongation from equilibrium, or $F = -k\Delta x$. A known force may be applied to balance the spring force by suspending a series of weights from the spring.

For example: 100g, 150g, 200g, 250g and 300g determination of the force of gravity in N and comparison with the

elastic force by registering the elongation of the spring Δx

Follow the instructions as described below

1. Install the spring and mark the end of the spring as x = 0, hang a known weight (Figure 1). and measure the elongation Δx. Record the results in **table 1**

2. Determine the spring constant k comparing the force of gravity with the force of elasticity:

$mg = k\Delta x$

$k = mg / \Delta x$

3. Connect the motion detector and place it directly under the hanging weight. Protect the motion detector by placing e.g. a wire basket over the detector. The mass should be approximately 30 cm above the detector when it is at rest.

4. Start by stretching the spring so that the weight moves up and down, vibrating about 10 - 15 cm between the minimum and maximum positions,

Be careful that the weight does not swing sideways

5. Record the data on the weight's position and speed.

Table 1

mass [kg]	weight Q [N]	elongation Δx [m]	spring constant k [N/m]

The mean value of the constant k = ...

Hardware required

- Vernier Go motion- motion detector
- Vernier Labquest mininterface (microcontroller) (instead of above you can Pico microcontroller with MicroPython and HCSR04P ultrasonic sensor)
- Pc or Mac

Materials required

- springs
- mass

Software required

- Vernier Logger Pro or Graphical Analysis (or Thony App if you use Pico)

Theory:

Energy is present in three forms for the mass of the sinker and the spring. A mass m, moving at a speed v, may have the kinetic energy Ek

Ek = 1 / 2mv2

The spring can store elastic potential energy Eps or gravity potential energy Ep. The elastic potential energy Eps can be calculated from the formula:

Eps = 1 / 2k Δx2, where k is the spring constant and Δx is the elongation or compression of the spring measured from the equilibrium position.

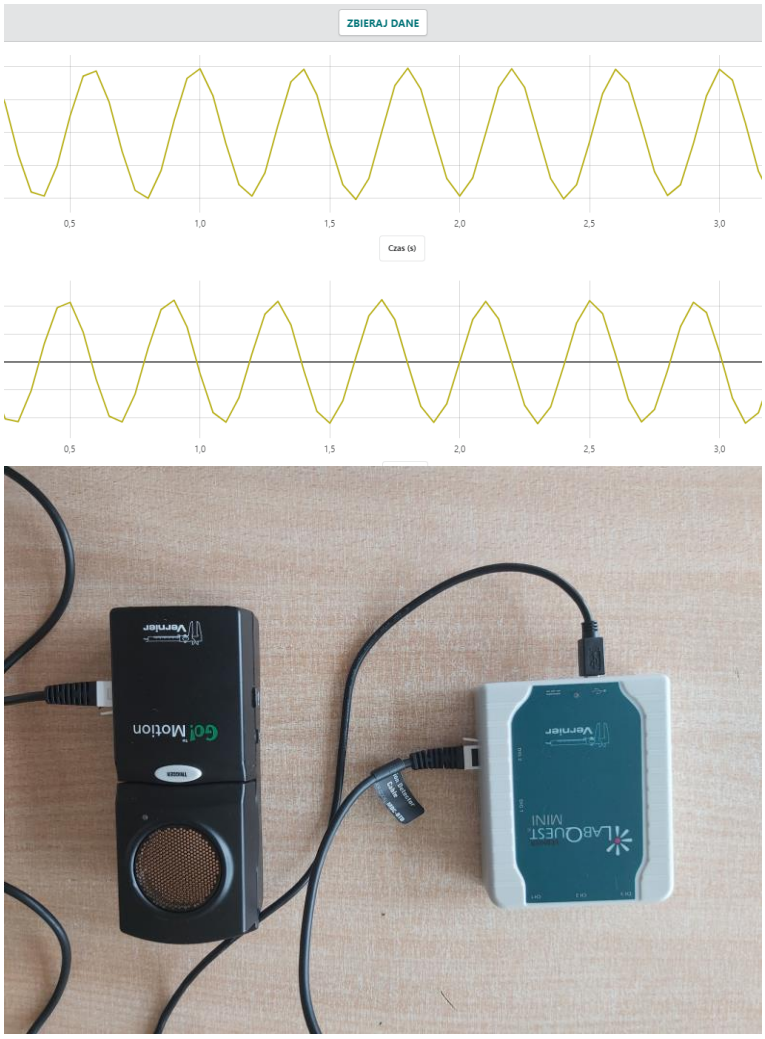
The weight-spring system also has gravitational potential energy (Ep = mgΔx), but we do not need to account for this energy if we measure the length of the spring from an equilibrium position. Then we can focus on the energy exchange between the kinetic energy and the elastic potential energy

If the system does not experience other forces, the principle of conservation of energy tells us that sum

Ek + Eps = 0, which we can test experimentally.

Experiment Evaluation

System testing:



Topic: Energy Introduction to the use of the Pico microcontroller in experiments. LED control. Thonny App.	Age >12	Country Poland	Date October 2021

Function, realisation

Introducing the use of a microcontroller to perform experiments. Getting to know the Thonny environment, controlling the pin states of the microcontroller on the example of LED

Hardware required

- Microcontroller Raspberry Pi Pico with Micropython 1.18 or above
- Cytron Maker Pico docking station for Pico
- PC or Mac computer.

Materials required

- 3xLED module (Dfrobot) with different colour
- 3xGrove male wire
- paper for note

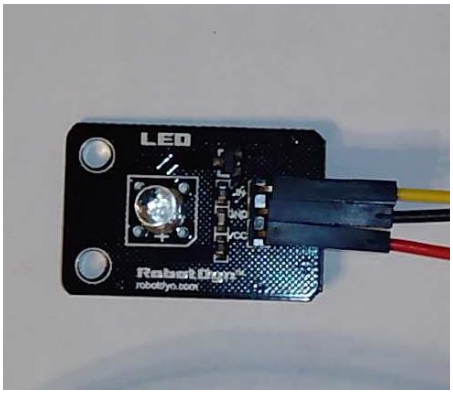
Software required

- ThonnyApp (thonny.org),
- this example don't need custom libraries

Setup Hardware

LED modules have 3 Pins: VCC, GND and OUT. Connect each LED according to the table below.

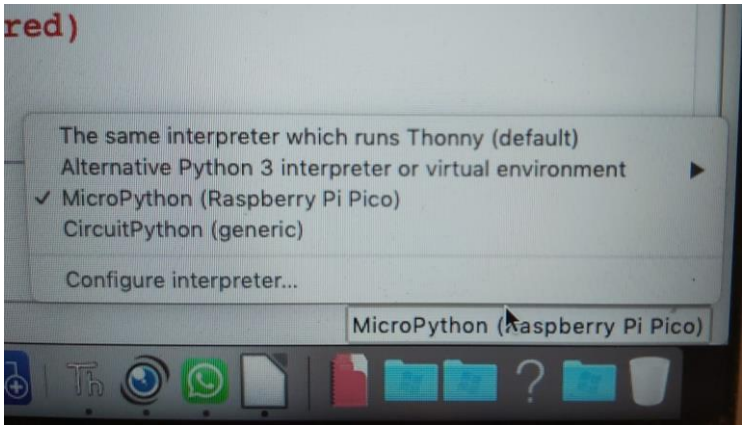
LED module	Raspberry Pi Pico pins	Grove male wire
VCC	3V3	red
GND	GND	black
Out (Signal)	GP3/GP5/GP9 (for 3 led modules)	yellow
not connected		white



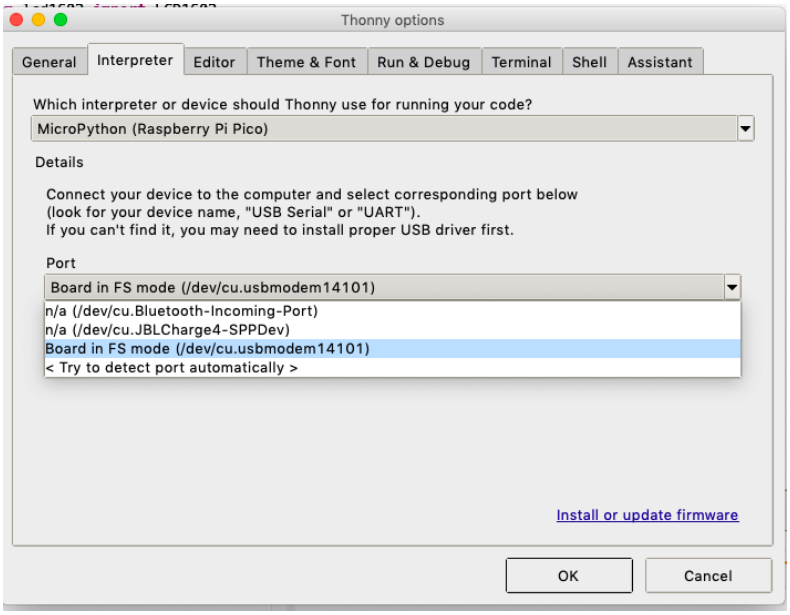
Setup software

In the lower right corner of the application window, select the Micropython Raspberry Pi Pico interpreter. In this case the code may bu run by Pico.

Sometimes the Pico board port will not be recognized automatically and you will have to manually point the interpreter and USB port. Select Menu "Tools/ Options"



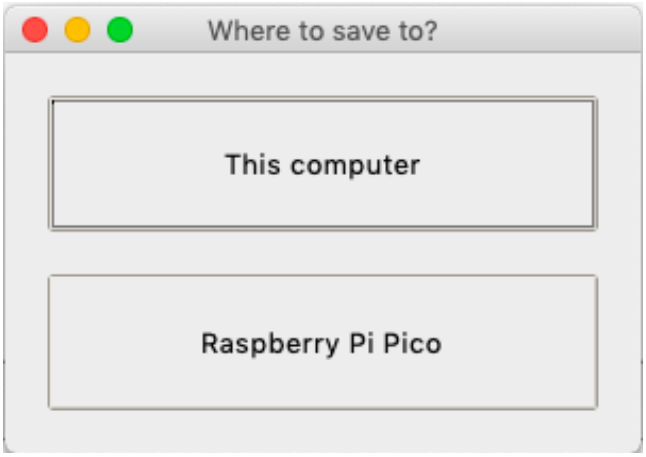
You can also use this screen to install the



MicroPython on the Pico board

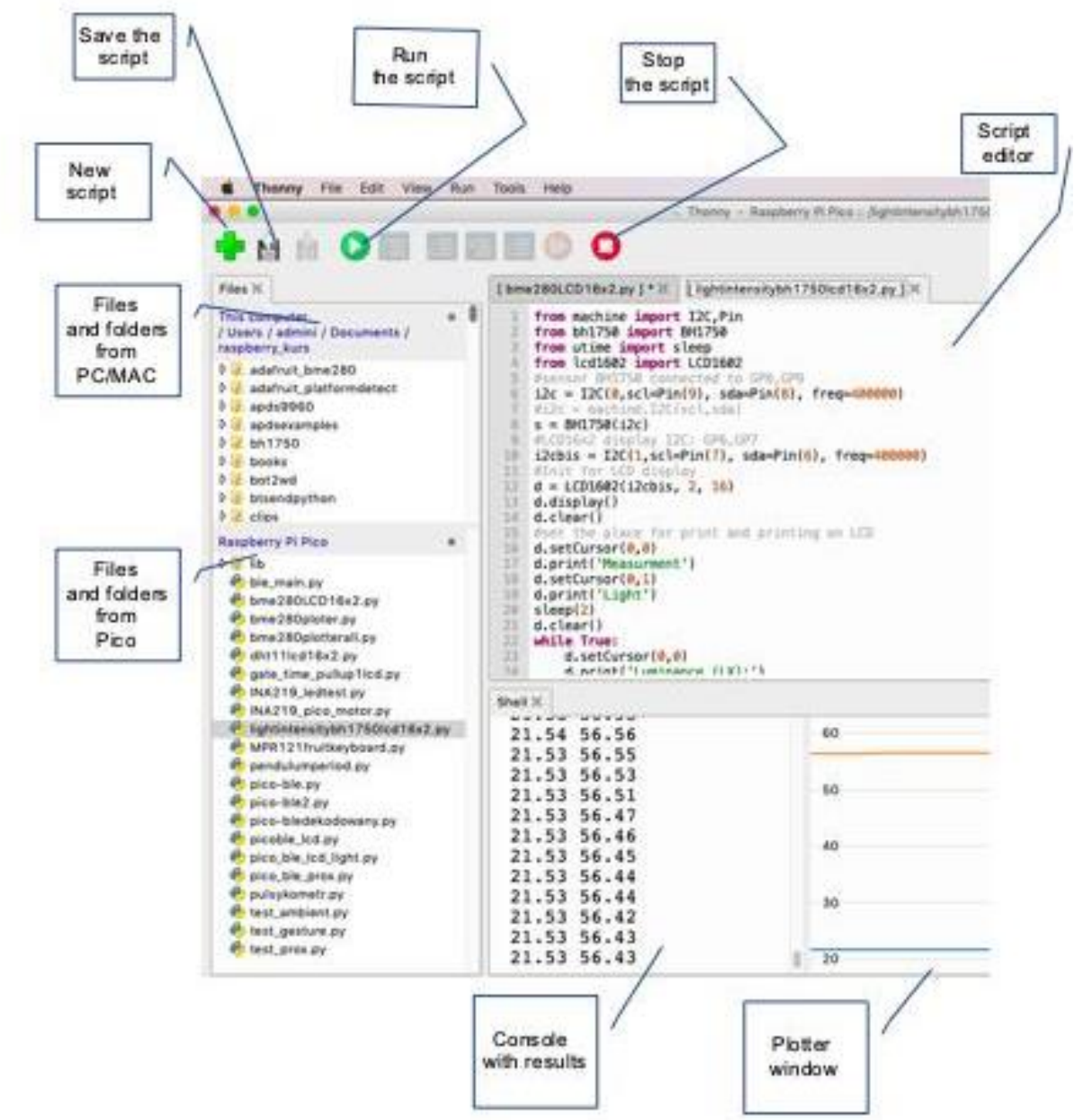
In this manual you can find simple Python code for saving in microcontroller using Thonny App. Write the script in Thonny editor, then select "File/Save" and select Raspberry Pi Pico as destination. Remember please to add .py at the end of filename. Our proposal for filename: LEDon.py.

After selecting the save option from the menu, you will be asked whether to save the script to local disk or Pico memory.



It is recommended to select "View / files" from the menu to see the scripts saved on the local disk and in the Pico memory.

An explanation of the Thonny application interface elements:



The digital pins of the microcontroller can be high (1) or low (0).
The led1.value (1) instruction sets the state of the pin to which the LED is connected to high.

Python code example to control 4 LEDs

```
from machine import Pin
from utime import sleep
# Pin numbers depend on connection with Cytron docking station
# for Cytron may be GP3,GP5,GP9
#onboard led internal connected to GP25
#grove wire connect to LED module (yellow to signal)
LED1= Pin(3,Pin.OUT)
LED2= Pin(5,Pin.OUT)
LED3= Pin(9,Pin.OUT)
LED0= Pin(25,Pin.OUT)
while True:
    #switch off all leds for 1 second
    LED1.value(0)
    LED2.value(0)
    LED3.value(0)
    LED0.value(0)
    sleep(1)
    #switch on all leds for 3 seconds
    LED1.value(1)
    LED2.value(1)
    LED3.value(1)
    LED0.value(1)
    sleep(3)
    #Switch off LED3 and internal led (GP25)
    LED3.value(0)
    LED0.value(0)
    sleep(4)
    #Switch off: LED1, LED2 and switch on: LED3 for 3 seconds
    LED1.value(0)
    LED2.value(0)
    LED3.value(1)
    LED0.value(1)
    sleep(3)
```

Example Code explanation

First two lines of the script:

```
from machine import Pin
from utime import sleep
```

mean the use of modules and functions defined in them
.

The **machine** module contains specific functions related to the hardware on a particular board.

The **utime** module (line 2) provides time functions. In this case we use **sleep** method for delay.

The following lines define the way of connecting the LED to the corresponding pins of the microcontroller (GP3, GP5, GP9 respectively). GP25 is internally connected with Pico onboard LED.

```
LED1= Pin(3,Pin.OUT)
LED2= Pin(5,Pin.OUT)
LED3= Pin(9,Pin.OUT)
LED0= Pin(25,Pin.OUT
```

The following line:

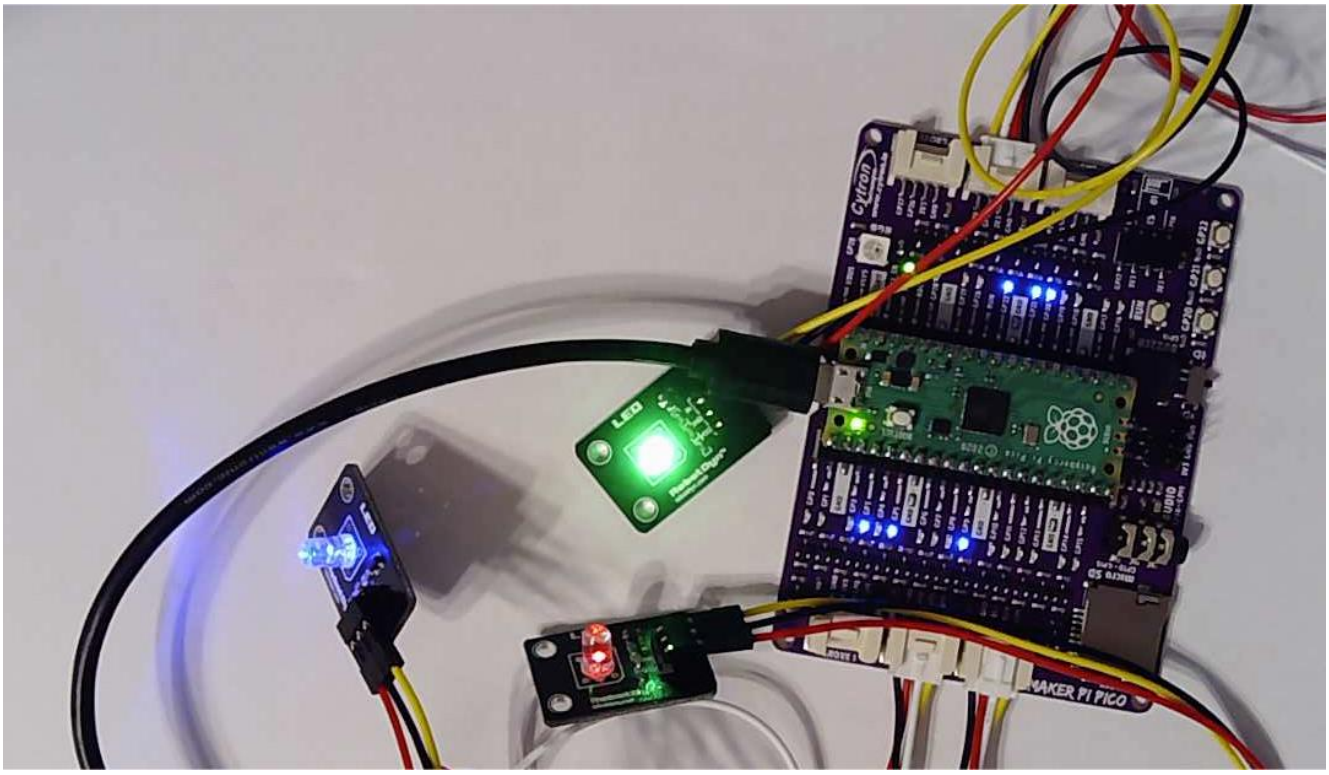
```
while True:
```

means infinite loop. The instructions in the loop will be repeated indefinitely. Pay attention to the indents. In the case of Python, they are very important.

Lines starting with # are comments.

Likewise, """remark""" text is a multi-line comment.

After writing the code inside editor window, we can run the code using the appropriate icon:



System operation effect

Go further

- Change the code and design a traffic light system at the crossroads.
- Use Pin 25 and onboard LED to have the microcontroller inform you about its condition in your projects.

Topic: All Resistance and voltage measurement (rotary analog resistor). Reading from analog sensor.	Age >14	Country Poland	Date October 2021

Function, realisation

Voltage and resistance measurement using a rotary potentiometer. Interpreting the measurement results from the analog sensor (ADC-Analog to Digital Converter).

Hardware required

- Microcontroller Raspberry Pi Pico with Micropython 1.18 or above
- Cytron Maker Pico docking station for Pico
- LCD16x2 Grove I2C 3V3
- PC or Mac computer.

Materials required

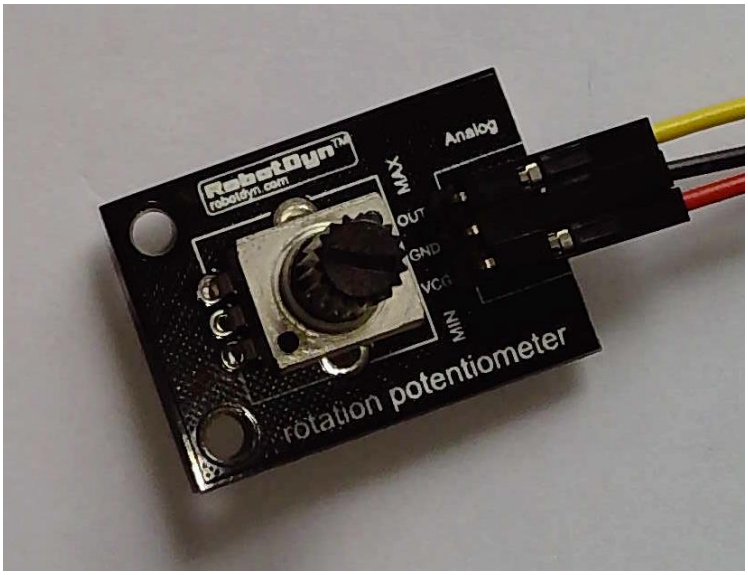
- 1xGrove standard wire (2xGrove grove plug)
- **1**xGrove male wire (1xGrove plug, 1xDuPont)
- 10 kΩ rotary potentiometer from Robotdyn
- paper for note

Software required

- ThonnyApp (thonny.org),
- simple version script does not require custom libraries
- LCD version require lcd1602 library (https://files.seeedstudio.com/wiki/Grove_Shield_for_Pi_Pico_V1.0/Libraries.rar)

Setup Hardware

10 kΩ rotary potentiometer have 3 Pins: VCC, GND and OUT. Connect potentiometer pins according to the table below:



Potentiometer	Raspberry Pi Pico pins	Grove male wire
VCC	3V3	red
GND	GND	black
Out (Signal)	GP27	yellow
not connected	GP26	white

LCD screen connection

LCD 1602 pins	Raspberry Pi Pico pins	Grove standard wire
GND	GND	black
VCC	3V3	red

SDA	**GP6**	**white**
SCL	**GP7**	**yellow**

Explanation of how the potentiometer works

When you rotate the knob, the sliding wiper connected to the knob slides on the resistive material with the rotation. With the sliding, the length of resistive material between U2– Uwyj and Uwyj– U1 changes, which causes the resistance value and thus the output voltage to change accordingly.

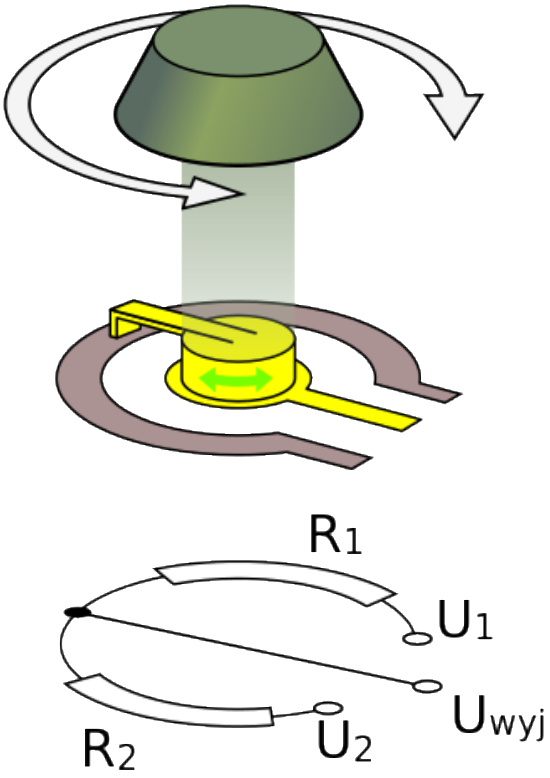


image source: pl.wikipedia.org

Reading an analogue input is almost identical to reading a digital input, except for one thing: when reading a digital input you use read(), but this analogue input is read with read_u16(). That last part, u16, simply warns you that rather than receiving a binary 0 or 1 result, you'll receive an unsigned 16-bit integer – a whole number between 0 and 65,535. If you can't get your Pico's analogue input to read exactly zero or exactly 65,535, that is correct. All electronic elements are built with a tolerance. In the case of the potentiometer, it will likely never reach exactly 0 or 100 percent of its input – but it will get you very close.

Setup software

Simple version (printing in console) of the script. Write the code inside Thonny App editor, save and run.

```
from machine import ADC
from utime import sleep
#ADC0,  GP26
#ADC1, signal from rotary resistor connected to GP27
potentiometer = ADC(27)
conversion_factor = 3.3 / (65535)
#Micropython ADC: 65536 steps from 0 to 65535, support 16 bit ADC
#Real ADC steps scale in Pico: 0-4095. Pico has only  12-bit ADC (levels: 0-4095)
#4095 correspond to 65535
print("RAW ADC reading, voltage:  ")
while True:
    raw=potentiometer.read_u16()
    voltage =raw  * conversion_factor
    print(raw,"\t",voltage)
    sleep(2)
```

Version with LCD support

```
from lcd1602 import LCD1602
from machine import I2C,Pin,ADC
from utime import sleep
#ADC0,  GP26
#ADC1, signal from rotary resistor connected to GP27
potentiometer = machine.ADC(27)
conversion_factor = 3.3 / (65535)
#Micropython ADC: 65536 steps from 0 to 65535, support 16 bit ADC
#Real ADC steps scale in Pico: 0-4095. Pico has only  12-bit ADC (levels: 0-4095)
#4095 correspond to 65535

#I2Cbus and LCD initialisation. LCD connected to GP6,GP7
i2c = I2C(1,scl=Pin(7),sda=Pin(6),freq=400000)
d = LCD1602(i2c,2,16)
d.display()
print("RAW ADC reading, voltage:  ")
while True:
    raw=potentiometer.read_u16()
    voltage =raw  * conversion_factor
    print(raw,"\t",voltage)
    d.setCursor(0,0)
    d.print('Raw ADC:'+str(raw))
    d.setCursor(0,1)
    d.print("Volts:"+str(round(voltage,3)))
    sleep(2)
    d.clear()
```

Example Code explanation

Reading voltage from analog Pin:

```
raw=potentiometer.read_u16()
```

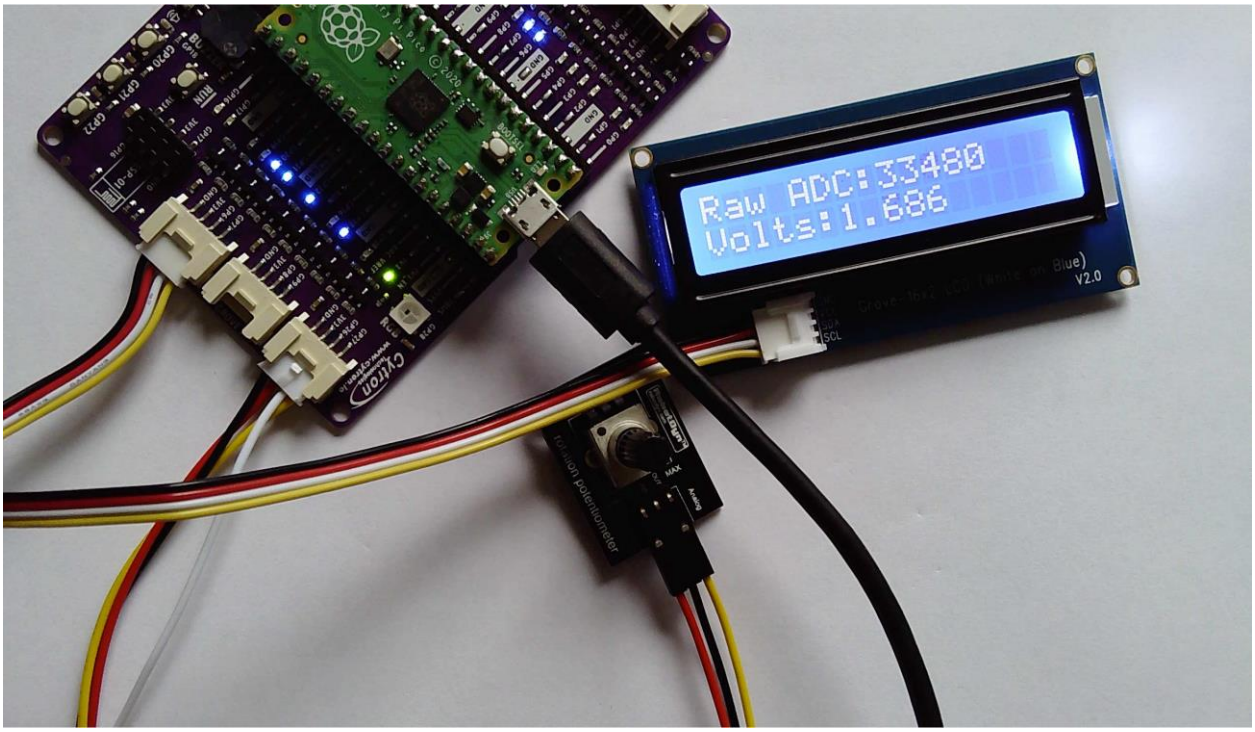
The number we see corresponds to the ADC level, in the range 0..65535. 65,535 means ‘full voltage-3.3V’, this is raw reading from ADC. We can easily change this to print the voltage corresponding to this level. W use conversion factor inside the script:

```
conversion_factor = 3.3 / (65535)
```

Then we can make calculations for voltage:

```
voltage =raw  * conversion_factor
```

System operation effect



Testing

Slowly turn the potentiometer knob. Observe the measurement results.

Note the results for the position of the knob:

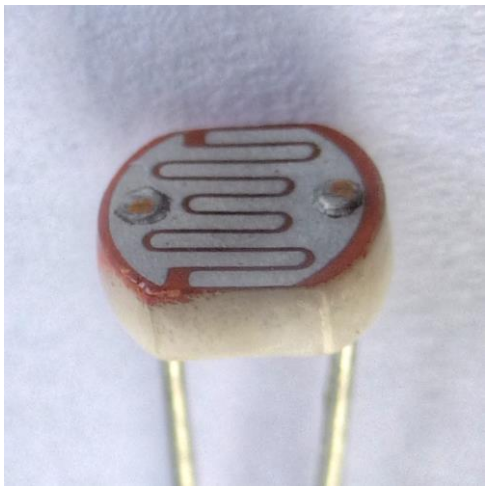
- turned all the way to the right,
- turned all the way to the left,

Try to reach voltage of 1.65V. What ADC level corresponds to 1.65V?

Go further

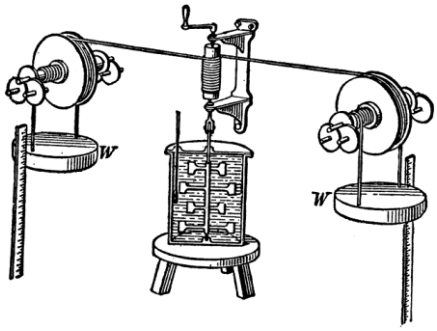
- instead of a potentiometer, use a photoresistor that changes the resistance depending on the light
- use a rain sensor instead of a potentiometer, the resistance of which changes as rain increases
- use an analog water quality sensor (TDS) instead of a resistor
- use the experience gained for other analog sensors. Compare the characteristics of the sensors and, on this basis, adjust the formula that calculates the sensor indications in the appropriate units

In the picture below you will find a photoresistor:



Topic	Age	Country	Date
Joule experiment revisited in modern ways	>14	Portugal	Oct 2021

Does agitating the water increase or decrease its temperature?



Joule´s experiments, that you have probably heard of, proves that temperature is related to the average of the kinetic energy of the particles composing the body. Let's check!

First, we must get the **temperature sensor working** (this is the technological part of the experiment). Use instructions in attachment if necessary.

Then, use the hand blender to agitate the water (very little water, just a bit in the bottom) and measure the temperature during agitation. ***Be sure the sensor is next to and not under the fans of the hand blender.***

Wait until water temperature is stable and the scale adjusts to a small interval (maybe from 19ºC to 20ºC). Turn on the hand blender. Observe and register the results.

Exploration:

1- *Does agitating the water increase or decrease temperature?*

2- *If we can warm up water both with heat and work, what is the connection between them?*

3- *If agitating the water increases the temperature, why does a very hot cup of tea cool off when we stir it?*

Extra Task:

Can you measure the efficiency of this heating process?

(Tip: use a chronometer to measure the time used and the power of the apparatus to establish the energy used in the process and $Q=m \cdot c \cdot \Delta T$ to measure the energy thewater has received. $c_{water}= 4{,}18\text{x}10^{3}\ J \cdot kg^{-1} \cdot °C^{-1}$)