

## **Data Acquisition Using the Raspberry Pi Pico W**

**Prof. David R. Loker, Pennsylvania State University**

David R. Loker received the M.S.E.E. degree from Syracuse University in 1986. In 1984, he joined General Electric (GE) Company, AESD, as a design engineer. In 1988, he joined the faculty at Penn State Erie, The Behrend College. In 2007, he became the Chair of the Electrical and Computer Engineering Technology Program. His research interests include wireless sensor networks, data acquisition systems, and communications systems.

# Data Acquisition Using the Raspberry Pi Pico W

## Abstract

The Raspberry Pi Pico W provides Wi-Fi capability and is the latest upgrade to the Pico product family. Similar to the basic Pi Pico, both boards utilize the same footprint, pinout and RP2040 microcontroller. They both include the same key I/O features, such as support for digital I/O (GPIO pins), PWM outputs, analog (ADC) inputs, and digital communication interfaces (SPI, I2C and UART). The one exception is that the Pico W incorporates the Infineon CYW43439 single chip radio which supports both 802.11n (2.4 GHz Wi-Fi) and Bluetooth 5.2 compatibility. Thus, the Pico W can be used in a wide variety of wireless applications.

Data acquisition is a common topic in courses that encompass both electrical and computer disciplines. The goal of this paper is to utilize the Pico W as an inexpensive alternative to the data acquisition hardware used for lab projects in electrical and computer engineering and engineering technology programs. First, lab projects are shown to introduce students to MicroPython (a small subset of the Python standard library), Thonny (the software development environment for writing Python code and downloading it to the Pico W), and API functions for peripheral control. Examples include LED control, analog input, OLED display, and a web server. Then, data acquisition projects are shown to illustrate the design of measurement systems. These include the design of a temperature measurement system and a digital voltmeter system. Projects contain engineering requirements, schematics, software code, and results. Then, an example of a student project utilizing data acquisition and Wi-Fi is provided.

## Introduction

Data acquisition (DAQ) involves sampling signals utilizing sensors that measure electrical parameters, processing these signals into real-world values, and displaying the information. This system is typically implemented with a USB DAQ device (e.g., myDAQ, etc.) connected to a PC, and software employed with graphical or text-based programming (e.g., LabVIEW, Matlab, etc.).

A variety of courses encompassing both electrical and computer disciplines involve data acquisition. One such course is Measurements and Instrumentation [1]. This reference describes a junior-level course, for an electrical and computer engineering technology program, which uses the myDAQ device for data acquisition, LabVIEW as the programming language, and a computer for the design and implementation of measurement systems. There are other courses that can also include data acquisition as a common topic. These include such courses as software programming, computer networking, communications systems, and project-based courses.

Embedded systems can provide an alternative to PC-based data acquisition hardware and software. Using an embedded system to read/analyze sensor data yields the ability to perform data acquisition at a higher rate than what the PC is able to do, and adhere to a strict time regiment, which may be necessary for data processing and analysis. Examples of courses that can use embedded systems include digital design, introduction to microprocessors, embedded systems design, intermediate microprocessors, and advanced microprocessors. Additionally,

there are courses which can use embedded systems as an integral component to the course. These courses can include wireless communications systems, control systems, and senior-level project-based courses.

An example of a hardware device for embedded systems is the inexpensive Raspberry Pi Pico W. MicroPython (a small subset of the Python standard library) can be used for software programming, and Thonny can be used as the software development environment for writing Python code and downloading it to the Pico W. The goal of this paper is to show how the Pico W can be used for data acquisition in student lab projects for electrical and computer engineering and engineering technology programs.

## MicroPython

MicroPython is a small subset of the Python standard library, and it is optimized to run on various microcontrollers (e.g., Pi Pico W, etc.) for embedded applications [2-4]. MicroPython provides built-in modules that are similar to the libraries found in Python [5]. Some examples include:

- time (functions: sleep, ticks\_ms, etc.)
- math (functions: sin, sqrt, pow, log, etc.)
- socket (methods: bind, listen, etc.)

MicroPython also contains libraries related to the hardware. Examples include:

- machine (classes: Pin, ADC, I2C, etc.)
- network (classes: WLAN, LAN, etc.)
- bluetooth (classes: BLE, UUID)

Additionally, there are MicroPython libraries designed specifically for peripherals connected to embedded microcontrollers. One such library is for the SSD1306 organic light emitting diode (OLED) display [6-7]. Various functions include fill, text, show, etc.

## Raspberry Pi Pico W

The Raspberry Pi Pico W board has the following key features [8].

- RP2040 microcontroller with 2MB of flash memory
- On-board single-band 2.4GHz wireless interfaces (802.11n, Bluetooth 5.2)
- Micro USB B port for power and data (and for reprogramming the flash)
- 3.3V general purpose I/O (GPIO)
  - 2 × UART
  - 2 × I2C
  - 2 × SPI
  - 16 × PWM channels
- 3 ADC inputs (12-bit 500ksps)
- RP2040 microcontroller (dual-core cortex M0+ at up to 133MHz)
- On-board USB1.1 (device or host)

Figure 1 contains the pinouts for the Raspberry Pi Pico W board. To interpret this diagram, it

should be noted that all of the I/O pins can operate as simple digital I/O (GPIO pins). In addition, complex functions such as SPI, I2C, UART and ADC are only available on specific pins.

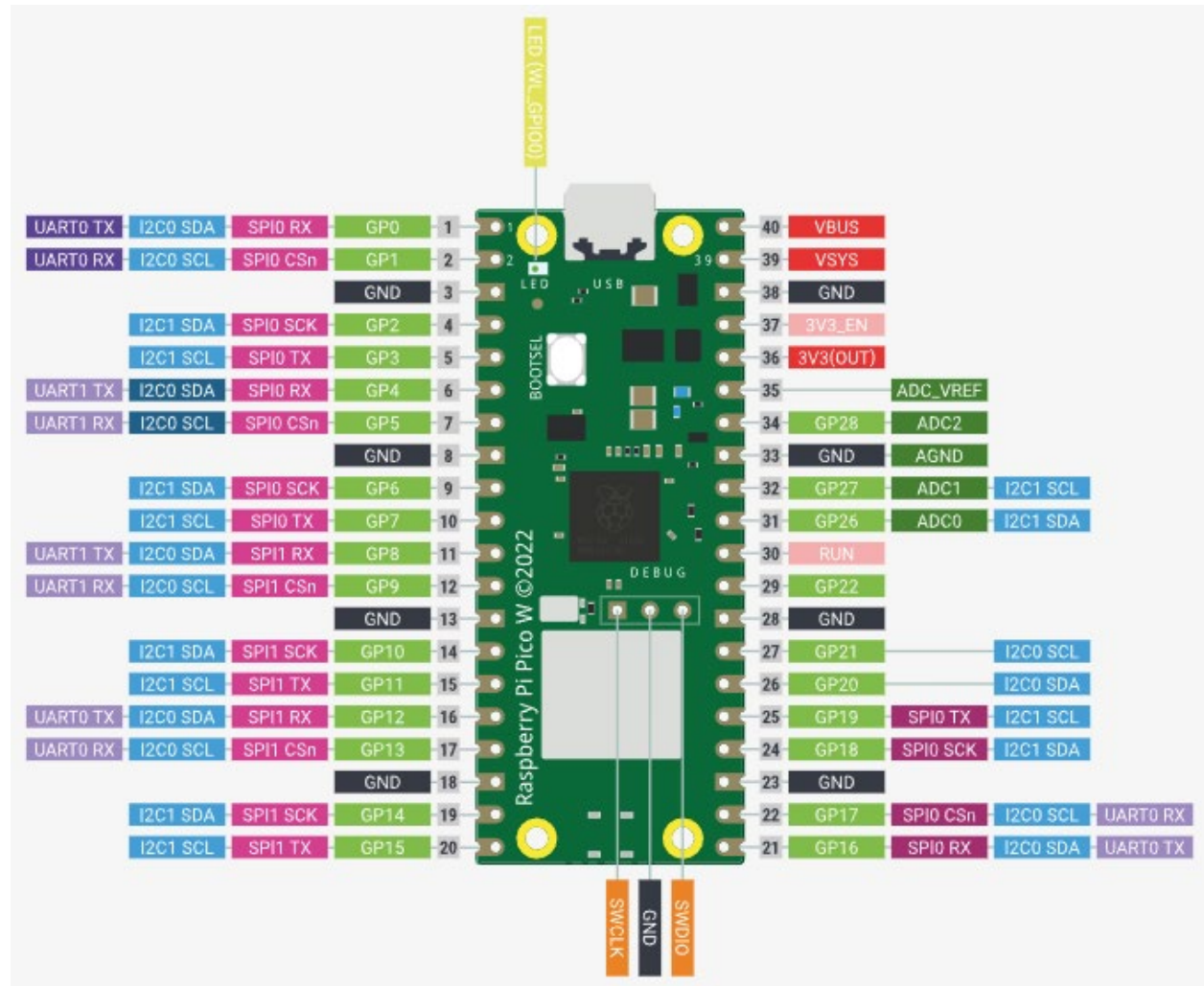


Figure 1. Pi Pico W Pinout [9]

### Introduction to MicroPython Coding Using Thonny

An example of an IDE for developing MicroPython code and downloading it to the Pico W is called Thonny [10]. It has a window area for writing code and a Python Shell for providing results to the console from running programs. An example is shown in Figure 2. This program creates a 1 kHz, 1V peak sinusoidal waveform from samples sufficient to produce 10 samples/cycle and 10 cycles. This results in a sample rate of 10 ksp/s and 100 total samples. The sinusoidal function is available from the math library. Mean and RMS values are displayed onto the console. On a Pico W, the console is simulated, and only available when running Thonny. In a stand-alone environment (no PC), the MicroPython “print” statement does not result in any text getting displayed to the user.

```
sin_volt_values.py x
1 import math
2
3 input_freq = 1e3
4
5 num_samples_per_cycle = 10
6 sample_freq = num_samples_per_cycle * input_freq
7
8 num_cycles = 10
9 num_samples = num_cycles * sample_freq/input_freq
10
11 volt_sum = 0
12 volt_sq_sum = 0
13 for i in range(num_samples):
14     sample_time = i * 1/sample_freq
15     volt = math.sin(2.0 * math.pi * input_freq * sample_time)
16     volt_sq = volt**2
17     volt_sum += volt
18     volt_sq_sum += volt_sq
19
20 volt_avg = volt_sum/num_samples
21 volt_rms = math.sqrt(volt_sq_sum/num_samples)
22
23 print("AVG volt: ", round(volt_avg,4))
24 print("RMS volt: ", round(volt_rms,4))

Shell x
MicroPython v1.23.0-preview.33.g2ed976f14 on 2024-01-08; Raspberry Pi Pico W
with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

AVG volt:  -0.0
RMS volt:  0.7071
```

Figure 2. Average and RMS Values for Sinusoidal Waveform

## LED Control

An example for controlling the on-board LED on the Pico W is shown in Figure 3. This program turns the LED on and off every 0.5 sec, and it also prints to the console the state of the LED. The program uses the Pin class (from the machine library) to control the on-board LED. Timing is controlled from the sleep function.

```
OnBoardLED.py ×
1 import machine
2 import time
3
4 led = machine.Pin("LED", machine.Pin.OUT)
5
6 while True :
7     led.on() # turns on the led
8     print("LED on")
9     time.sleep(.5)
10    led.off() # turns off the led
11    print("LED off")
12    time.sleep(.5)

Shell ×
MicroPython v1.23.0-preview.33.g2ed976f14 on 2024-01-08;
Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

LED on
LED off
LED on
LED off
```

Figure 3. Control of On-Board LED

## Analog Input

An LM35 Temperature Sensor is connected to ADC2 (pin 34 – GPIO28) on the Pi Pico W [11]. The LM35 has a temperature coefficient of 10mV/°C. The ADC has 3.3V full-scale with 12 bits of resolution. The 12-bit value is converted to 16 bits (i.e., the 0-4095 input range is represented as 0-65535). The MicroPython code and results are shown in Figure 4.

```
Temp_C_read_W.py ×
1 import machine
2
3 pin34 = machine.ADC(28)
4 data = machine.ADC.read_u16(pin34)
5 volts = 3.3 * data / 65535.0
6 print("Input is ", volts, " volts \n")
7 temp = volts / 0.010
8 print("Temp in degrees C is ", temp)

Shell ×
MicroPython v1.23.0-preview.33.g2ed976f14 on 2024-01-08;
Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Input is  0.2465873  volts

Temp in degrees C is  24.65873
```

Figure 4. Analog Input

## OLED Display

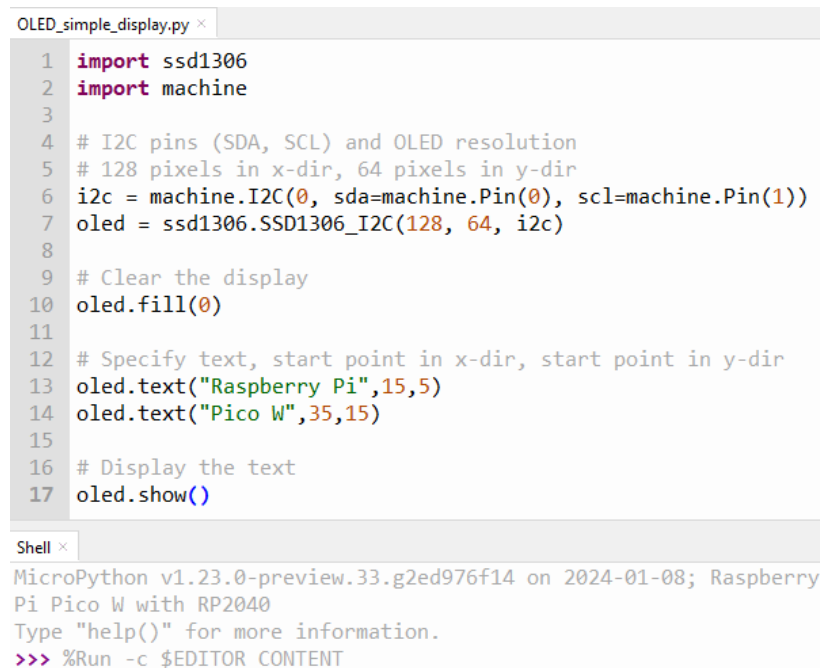
The MicroPython library file for the organic light emitting diode (OLED) display (SSD1306) is available for download [7]. For the I2C interface, the bus connections are SDA (Serial data) and SCL (Serial clock) [12]. Connections from the Pico W to the display are:

- I2C0 SDA (Pico W– Pin 1 – GPIO0) to SDA (SSD1306)
- I2C0 SCL (Pico W– Pin 2 – GPIO1) to SCL (SSD1306)
- GND (Pico W– Pin 8) to GND (SSD1306)
- 3V3(OUT) (Pico W– Pin 36) to VCC (SSD1306)

The OLED display is configured for 128x64 pixels. Software commands include:

- `fill(0)` *clear the display*
- `text("text_to_display",x_start,y_start)`
- `show()` *display the text*

The MicroPython code is shown in Figure 5, and an image of the display is shown in Figure 6.



```
OLED_simple_display.py ×
1 import ssd1306
2 import machine
3
4 # I2C pins (SDA, SCL) and OLED resolution
5 # 128 pixels in x-dir, 64 pixels in y-dir
6 i2c = machine.I2C(0, sda=machine.Pin(0), scl=machine.Pin(1))
7 oled = ssd1306.SSD1306_I2C(128, 64, i2c)
8
9 # Clear the display
10 oled.fill(0)
11
12 # Specify text, start point in x-dir, start point in y-dir
13 oled.text("Raspberry Pi",15,5)
14 oled.text("Pico W",35,15)
15
16 # Display the text
17 oled.show()

Shell ×
MicroPython v1.23.0-preview.33.g2ed976f14 on 2024-01-08; Raspberry
Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
```

Figure 5. OLED Display Code



Figure 6. OLED Display Image

## Web Server

The Pico W contains an onboard 2.4 GHz wireless interface using the Infineon CYW43439 (802.11n) [13-14]. The network library allows a connection to a Wi-Fi network with the specified access point name (ssid) and password [15-16]. Then, a socket is opened (port 80) to allow a web browser to access the web server. The MicroPython code is shown in Figure 7. The code to create the webpage for the web server is shown in Figure 8. The webpage consists of a title and the main content containing text and the temperature from an LM35 temp sensor. The *get\_LM35\_temperature()* function uses ADC2 (pin 34 – GPIO28) to read the voltage and convert the temp to °C (see Figure 4). An image of the webpage is shown in Figure 9. This shows the PC connected to IP address 192.168.137.28. Once the Pico W is connected to the local Wi-Fi access point, an IP address is assigned and displayed on the Thonny console.



```
WiFi_temp_display_new.py ×
1 import network
2 import socket
3 import time
4 import machine
5
6 # B148 access point
7 ssid = 'XXXXXXXXXX'
8 password = 'XXXXXXXXXX'
9
10 def connect():
11     # Connect to WLAN
12     wlan = network.WLAN(network.STA_IF)
13     wlan.active(True)
14     wlan.connect(ssid, password)
15     while not wlan.isconnected():
16         print('Connecting...')
17         print(wlan.status())
18         time.sleep(1)
19     ip = wlan.ifconfig()[0]
20     print(f'Connected on {ip}')
21     return ip
22
23 def open_socket(ip):
24     # Open a socket
25     address = (ip, 80)
26     connection = socket.socket()
27     connection.bind(address)
28     connection.listen(1)
29     return connection
```

Figure 7. Wi-Fi Network Access

```

43 def webpage(temperature):
44     # Template HTML
45     html = f"""
46         <!DOCTYPE html>
47         <html>
48         <head>
49             <title>Raspberry Pi Pico W Project</title>
50         </head>
51         <body>
52             <p>This project utilizes the Pico W's ability to host a web server to monitor temperature using an LM35.</p>
53             <p>Temperature is {temperature}</p>
54         </body>
55         </html>
56     """
57     return str(html)
58
59 # Start a web server
60 def serve(connection):
61     temperature = 0
62
63     while True:
64         client = connection.accept()[0]
65         request = client.recv(1024)
66         request = str(request)
67
68         try:
69             request = request.split()[1]
70         except IndexError:
71             pass
72
73         temperature = get_LM35_temperature()
74
75         html = webpage(temperature)
76         client.send(html.encode() if html else b"")
77         client.close()
78
79     try:
80         ip = connect()
81         connection = open_socket(ip)
82         serve(connection)
83     except KeyboardInterrupt:
84         machine.reset()

```

Figure 8. Web Server Code

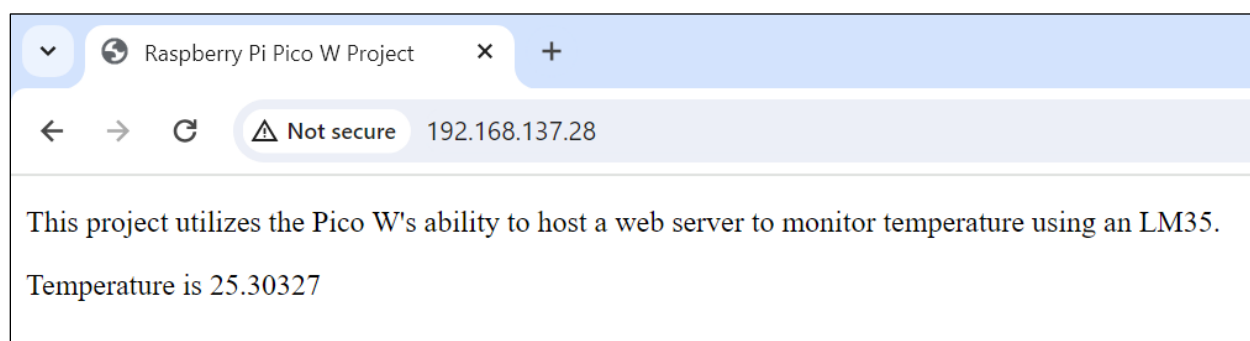


Figure 9. Webpage

## Data Acquisition Projects

There are two data acquisition projects presented in this paper. These include a temperature measurement system and a digital voltmeter system.

## Temperature Measurement System

The engineering requirements for the temperature measurement system are shown below.

- LM35 used as the temperature sensor
- Temperature range from 0°C to 100°C
- Temperature accuracy is within +/- 5°C of temperature standard
- Data collection:
  - One voltage reading every 0.1 seconds
  - Total of ten voltage readings collected
- Output is the average temperature in °C for one second

Since the ADC has a full-scale voltage of 3.3V with 12 bits of resolution, the voltage resolution (i.e., step size) is 0.806mV/count. With a temperature coefficient of 10mV/°C for the LM35, the full-scale temperature is 330°C and the temperature resolution is 0.0806°C/count. Temperature resolution can be improved by using a signal conditioning circuit at the output of the LM35 to scale the voltage such that the full voltage of the ADC is used for the complete temperature range. This would result in a temperature resolution of 0.0244°C/count. The MicroPython code implementing the temperature measurement system is shown in Figure 10.

Testing was performed as the LM35 temp sensor, mounted on a breadboard, was heated by using a hair dryer. Results from the Pico W were compared to a Fluke 87 V DMM with a type-K thermocouple probe used as the temperature standard. The thermocouple was placed in direct contact with the LM35 (TO-92 plastic package) as the sensor was heated. The results are shown in Table 1. Worst-case temperature error was 1.27°C or 4.32%, which may be primarily due to the procedure for heating the sensor.

```
Input_volt_avg_temp.py ×
1 import machine
2 import math
3 import time
4
5 pin34 = machine.ADC(28)
6
7 num_samples = 10
8 volt_sum = 0
9 for i in range(num_samples):
10     data = machine.ADC.read_u16(pin34)
11     volts = 3.3 * data / 65535.0
12     volt_sum += volts
13     time.sleep(0.1)
14
15 volt_avg = volt_sum/num_samples
16 print("AVG volt: ", round(volt_avg,4))
17 temp = volt_avg / 0.010
18 print("Temp in degrees C is ", round(temp,4))
```

Figure 10. Temperature Measurement System

Table 1. Temperature Measurement System Results

Temp in °C		Temp Error	
Pico W	Fluke 87 V	°C	%
30.6691	29.4	1.2691	4.3167
45.9739	46.3	-0.3261	-0.7043
56.5363	56.2	0.3363	0.5984
80.4861	80.6	-0.1139	-0.1413

### Digital Voltmeter System

The engineering requirements for the digital voltmeter system are shown below.

- Functional for sinusoidal, triangular, and square waveforms
- Waveform frequency range from 1 kHz to 10 kHz
- DC and AC RMS values displayed
- True RMS values determined
- Accuracy is within 5% of voltage using a digital scope as the standard
- Data collection:
  - Minimum of 10 samples/cycle over full frequency range
  - Minimum of 10 cycles over full frequency range

Based on the data collection requirement, both the sample rate ( $f_s$ ) and number of samples ( $s$ ) can be determined as follows.

- $f_s = f_{MAX} * 10 \frac{\text{samples}}{\text{cycle}} = 100\text{ksp}$
- $\frac{\text{samples}}{\text{cycle}}(MAX) = \frac{f_s}{f_{MIN}} = \frac{100\text{ksp}}{f_{MIN}} = 100 \frac{\text{samples}}{\text{cycle}}$
- $s = \frac{\text{samples}}{\text{cycle}}(MAX) * 10\text{cycles} = 1000$

For high-speed data acquisition, MicroPython code is implemented for the ADC using Direct Memory Access (DMA) [17]. The library file (rp\_devices.py) and sample test file (rp\_adc\_test.py) are available for download [18]. There are several variables in the sample test file that need changed.

- ADC\_CHAN = 2      # For ADC2
- NSAMPLES = 1000    # Number of samples
- RATE = 100000      # Sample rate of 100ksp

Additional code for determining DC and AC RMS values is placed at the bottom of the sample test file previously downloaded. This code is shown in Figure 11.

Testing was performed with sinusoidal, square, and triangular waveforms at voltages of 2.0Vpp and 1.5V DC offset. Data was collected at both 1kHz and 10kHz. Sampling parameters were constant at  $f_s=100\text{ksp}$  and 1000 samples. Results from the Pico W were compared to results measured with an Agilent DSO\_X 3012A scope. The time base on the scope was set to display 10 full cycles of the waveform. Measurement features of the scope were selected for AVG-FS (full screen DC value) and AC RMS-FS (full screen AC RMS value). Theoretical values for AC

RMS are calculated as follows.

- $v_{ACRMS} = \frac{v_{PK}}{\sqrt{2}}$  (*sinusoidal*)
- $v_{ACRMS} = v_{PK}$  (*square*)
- $v_{ACRMS} = \frac{v_{PK}}{\sqrt{3}}$  (*triangular*)

Results are shown in Table 2. Percentage error was calculated based on using the Agilent DSO\_X 3012A scope as the standard. All percentage errors were less than 5%. For the DC results, errors ranged from 3.68% to 4.89%. For the AC RMS results, errors ranged from -0.74% to 0.60%. More accurate results for the Pico W may be obtained from higher sampling rates with more data collected.

```
rp_adc_DAQ_waveform.py x
72 volt_sum = 0
73 volt_sq_sum = 0
74 volt = []
75 for i in vals:
76     volt=(float(i))
77     # print(volt)
78     volt_sq = volt**2
79     volt_sum += volt
80     volt_sq_sum += volt_sq
81
82 volt_avg = volt_sum/NSAMPLES
83 volt_rms = math.sqrt(volt_sq_sum/NSAMPLES)
84
85 print("AVG volt: ", round(volt_avg,4))
86 print("DC RMS volt: ", round(volt_rms,4))
87
88 ac_volt_sq_sum = 0
89 ac_volt = []
90 for i in vals:
91     ac_volt=(float(i))-volt_avg
92     # print(ac_volt)
93     ac_volt_sq = ac_volt**2
94     ac_volt_sq_sum += ac_volt_sq
95
96 ac_volt_rms = math.sqrt(ac_volt_sq_sum/NSAMPLES)
97
98 print("AC RMS volt: ", round(ac_volt_rms,4))
```

Figure 11. DC and AC RMS Code

Table 2. Digital Voltmeter Results

Waveform	Freq	Pico W		Scope		% Error	
		DC	AC RMS	DC	AC RMS	DC	AC RMS
Sinusoidal	1kHz	1.505	0.7124	1.4488	0.7124	3.8791	0.0000
Sinusoidal	10kHz	1.5197	0.7087	1.4488	0.7136	4.8937	-0.6867
Square	1kHz	1.509	1.0112	1.4476	1.009	4.2415	0.2180
Square	10kHz	1.5087	1.0028	1.4476	1.0103	4.2208	-0.7424
Triangular	1kHz	1.5021	0.5847	1.4488	0.5812	3.6789	0.6022
Triangular	10kHz	1.5164	0.5846	1.4488	0.5823	4.6659	0.3950

### Student Project

An individual project is required for a 1-credit senior-level seminar course [3]. An example of a student project is shown in Figure 12. This project utilizes the Wi-Fi module on the Pico W to create a web server for monitoring and controlling an electrical system. A schematic for the system is included in Figure 13. The web browser provides the user interface for remotely interacting with the electrical system. An example of the web browser is illustrated in Figure 14. Current status of the system is also emailed, as indicated in Figure 15.

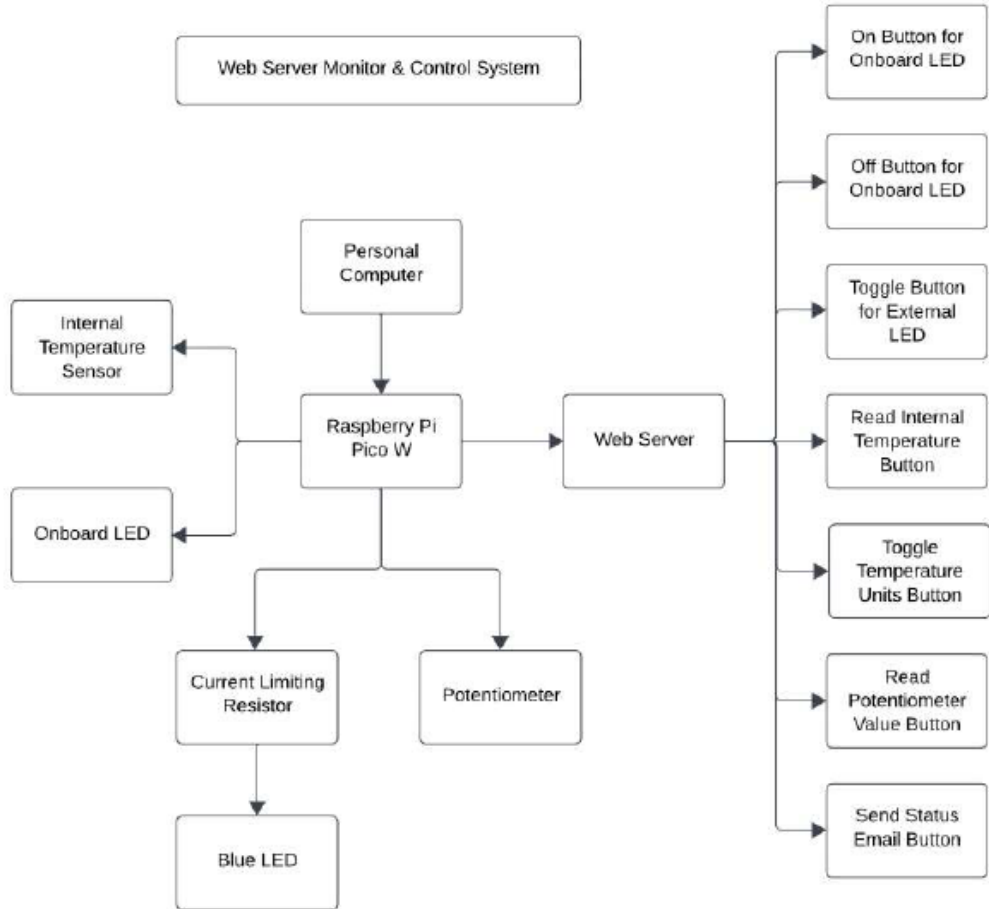


Figure 12. Student Project Block Diagram

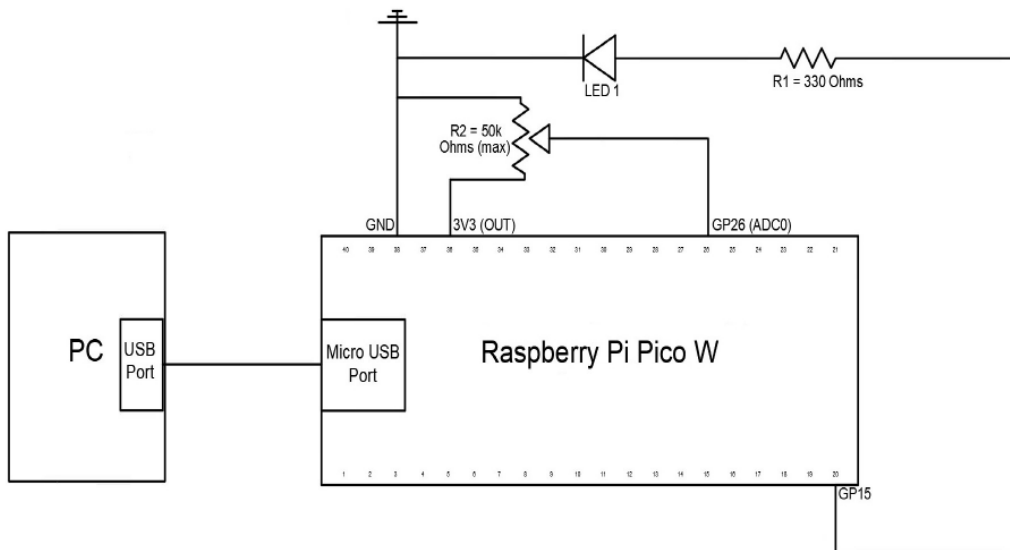


Figure 13. Student Project Schematic

## Pico W Project

This project utilizes the Pico W's ability to host a web server to control and monitor various components. It also utilizes SMTP to email the status of the indicators.

### Controls

Turn On Onboard LED	Turns on the onboard LED.
Turn Off Onboard LED	Turns off the onboard LED.
Toggle External LED	Toggles the state of the external LED.
Read Temperature	Reads the current temperature.
Toggle Display (C/F)	Toggles the temperature display between Celsius and Fahrenheit.
Read Potentiometer	Reads the resistance of the potentiometer.
Send Status Email	Sends the current status via email.

### Indicators

Onboard LED is OFF	External LED is OFF	Temperature is 22.36°C	Potentiometer Resistance is 24334 ohms
--------------------	---------------------	------------------------	--

12:54:12 Fri Dec 08 2023

Figure 14. Student Project Webpage

Pico W Status Update Inbox x

**T** Pico W [redacted] >  
to bcc: me ▾

Status Update from Pico W  
Onboard LED is ON  
External LED is ON  
Temperature is 22.36°C  
Temperature is 72.25°F  
Potentiometer Resistance is 24334 ohms

Figure 15. Student Project Status Email

## Student Assessment

Table 1 provides results from a student questionnaire regarding the lab project. One student completed the questionnaire, and this is the student that completed the project shown in Figures



12-15. This student worked approximately 20-25 hours on the project and found the level of difficulty for using the Pico W and Thonny to be easy, online resources to be easy to find, and the level of interest in using the Pico W on future projects to be positive.

Additional student feedback on using the Pico (without the Infineon CYW43439 device) and Thonny for lab projects, was performed [3]. These same projects can be implemented on the Pico W, with the addition of Wi-Fi and Bluetooth capability. The results of this assessment indicated significant student interest in using the Pico.

Table 1. Student Questionnaire Results.

Questions	Possible Responses
1. Approximate time in hours you worked on your project. (e.g., 1-10, 11-20, 21-30, 31-40, >40)	<ul style="list-style-type: none"> <li>• 21-30</li> </ul> <p>20-25 hours most likely</p>
2. Level of difficulty for using the Pico W. (e.g., easy, moderately difficult, extremely difficult)	<ul style="list-style-type: none"> <li>• 1 (easy)</li> </ul>
3. Level of difficulty for using Thonny. (e.g., easy, moderately difficult, extremely difficult)	<ul style="list-style-type: none"> <li>• 1 (easy)</li> </ul>
4. Availability of online resources for using the Pico W. (e.g., easy to find, moderately difficult to find, extremely difficult to find)	<ul style="list-style-type: none"> <li>• 1 (easy to find)</li> </ul> <p>A lot of the regular Pico's resources still apply.</p>
5. What is your level of interest in using the Pico W for future projects? (e.g., positive, neutral, negative)	<ul style="list-style-type: none"> <li>• Positive</li> </ul> <p>I would have loved another class to explore or build off of.</p>
6. Provide at least two learning outcomes from using the Pico W on this project.	<ol style="list-style-type: none"> <li>1. Learned how to configure a web server for both host and clients.</li> <li>2. Learned how to use SMTP protocol to send circuit readings via email.</li> <li>3. Learned how to utilize the Pico W's ADC to read internal &amp; external temp and resistance.</li> </ol>
Any other comments?	<ul style="list-style-type: none"> <li>• The Pico W is only a few years old, so new resources are still popping up frequently.</li> <li>• The Pico W may be small, but it offers a lot of versatility in its functionality. It made it easy to adapt my project as challenges came up.</li> </ul>

## Summary

The Raspberry Pi Pico W provides Wi-Fi capability and is an inexpensive board suitable for many applications in a variety of courses. MicroPython is optimized to run on various microcontrollers, including the Pico W board. Thonny can be used as the Python IDE, and it is available for free to be used as the software development environment.

Several lab projects were presented to introduce some of the capabilities of the Pico W. These included on-board LED control, reading an analog input voltage, using an OLED display, and designing a web server. Then, two data acquisition projects were designed using the Pico W: temperature measurement system and digital voltmeter system. Additionally, a student project was shown. Overall feedback from the student regarding the project was very positive, with an expressed interest in having another class utilizing the Pico W on projects.

Preliminary results are promising and indicate that the Pico W with MicroPython could be used as an inexpensive embedded alternative to PC-based data acquisition systems typically implemented with a USB DAQ device (e.g., myDAQ, etc.) and software employed with graphical or text-based programming (e.g., LabVIEW, Matlab, etc.). More testing with data acquisition projects and student feedback will be needed to confirm these conclusions.

## References

- [1] D. Loker and S. Strom, "Innovative Laboratory Projects for a Measurements and Instrumentation Course," *Annual Meeting, American Society for Engineering Education*, 2019.
- [2] D. Loker, "MicroPython in a Wireless Communications Systems Course," *Annual Meeting, American Society for Engineering Education*, 2021.
- [3] D. Loker and S. Strom, "Embedded Systems using the Raspberry Pi Pico," *Annual Meeting, American Society for Engineering Education*, 2022.
- [4] D. Loker, "Raspberry Pi Pico as an IoT Device," *Annual Meeting, American Society for Engineering Education*, 2023.
- [5] MicroPython.org. [Online]. Available: <https://docs.micropython.org/en/latest/>
- [6] MakerPortal.com. [Online]. Available: <https://makersportal.com/shop/ssd1306-oled-display-kit>
- [7] GitHub.com. [Online]. Available: <https://github.com/makerportal/rpi-pico-ssd1306/tree/main/micropython>
- [8] Raspberrypi.com. [Online]. Available: <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>
- [9] Raspberrypi.com. [Online]. Available: <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>
- [10] Thonny.org. [Online]. Available: <https://thonny.org/>
- [11] ti.com. [Online]. Available: [https://www.ti.com/lit/ds/symlink/lm35.pdf?ts=1707115526692&ref\\_url=https%253A%252F%252Fwww.mouser.de%252F](https://www.ti.com/lit/ds/symlink/lm35.pdf?ts=1707115526692&ref_url=https%253A%252F%252Fwww.mouser.de%252F)
- [12] SparkFun.com. [Online]. Available: <https://learn.sparkfun.com/tutorials/i2c/all>
- [13] RaspberryPi.com. [Online]. Available: <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>
- [14] ieee.org. [Online]. Available: <https://standards.ieee.org/ieee/802.11n/3952/>
- [15] RaspberryPi.com. [Online]. Available: <https://datasheets.raspberrypi.com/picow/connecting-to-the-internet-with-pico-w.pdf>
- [16] RaspberryPi.com. [Online]. Available: <https://projects.raspberrypi.org/en/projects/get-started-pico-w/2>
- [17] iosoft.blog. [Online]. Available: <https://iosoft.blog/2021/10/26/pico-adc-dma/>
- [18] github.com. [Online]. Available: <https://github.com/jbentham/pico>