

Python-based Microcontroller Architecture and Microcontroller Application Education in Engineering Technology

Dr. Byul Hur, Texas A&M University

Dr. B. Hur received his B.S. degree in Electronics Engineering from Yonsei University, in Seoul, Korea, in 2000, and his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Florida, Gainesville, FL, USA, in 2007 and 2011, respectively. In 2016, he joined the faculty of Texas A&M University, College Station, TX. USA, where he is currently an Associate Professor. His research interests include Mixed-signal/RF circuit design and testing, measurement automation, environmental & biomedical data measurement, and educational robotics development.

Python-based Microcontroller Architecture and Microcontroller Application Education in Engineering Technology

Byul Hur

Department of Engineering Technology and Industrial Distribution
Texas A&M University, College Station

Abstract

Python gained good popularity in computer programming education. Generally, Python is a high-level language, and it is considered a scripting language. For firmware or microcontroller education, compiler-based C/C++ languages are typically taught in courses and lessons. As an alternative approach, Python script language can be used in creating programs to control microcontrollers and processors. In this paper, MicroPython will be reviewed and studied as a potential use in microcontroller and embedded system education. MicroPython is open-source software, and it is a lean version of a standard Python. In this paper, several programming laboratory examples will be presented, and they are written in MicroPython for a Raspberry Pi Pico, RP2040. Alternative MicroPython-based microcontroller programming methods can be potentially taught as additional topics as a part of the microcontroller application course curriculum course. The pros and cons of the MicroPython-based microcontroller application lessons and laboratory examples will be presented and discussed in this paper.

Introduction

Python is a high-level programming language, and it gained good popularity in computer programming education [1][2]. Generally, Python can be understood as a scripting language. Typically, for microcontroller education, students may learn C/C++ languages. In order to learn about low-level processor architecture and operations, assembly language can be also taught in class. In these C/C++ or Assembly languages, a source code written by programmers and students can be compiled to generate executable binary code for a processor. Next an executable binary can be transferred and stored in the program memory space for a microcontroller or a microprocessor.

As the popularity of the Python script language has increased, the lessons and classes for microcontroller courses using Python languages have been increasingly covered and adopted in education. To be specific, the name of the Python language for a microcontroller education environment is MicroPython [3]. It is a lean and efficient implementation of the Python programming language. Users can rapidly generate a program that controls a microcontroller using Python. MicroPython is an open-source software. There are variants such as CircuitPython [4]. A decent amount of traditional educational C/C++ programs can also be implemented using MicroPython. Some educators found this method to be an easy and fun programming environment for the students. The author teaches Assembly, C/C++, and Python-based microcontroller programming in his embedded system courses. Learning Python-based programming for microcontrollers can be a

useful technique for students. In this paper, a portion of MicroPython lessons and laboratory examples are presented. In these examples, a Raspberry Pi Pico, RP2040, board is used, and lesson examples are presented [5]. Moreover, the discussion about the pros and cons of the MicroPython-based microcontroller application course curriculum as well as the lessons learned through Python-based education are presented.

Microcontroller Education in Higher Education Institutions

Microcontroller courses are typically found to be offered in higher education institutions. Some microcontroller courses may focus on theoretical studies and designs. Some microcontroller courses may emphasize more on creating application systems. For the students' long-term gain, both of the aspects are important in microcontroller education. For instance, first, students may need to earn a good understanding of both hardware and software aspects through pre-requisite courses such as circuit courses and basic programming courses. Next, as another pre-requisite course, students may need to take courses that cover microcontroller/processor architecture. Then, it would be effective for them to take microcontroller application-focused courses to learn how to build custom electronics systems and embedded systems. However, there can be some self-taught students and hobbyists who might have skipped some fundamentals and theory, but they prefer to use an open-source platform such as Arduino to create electronic systems rapidly. These rapid prototypes created using open-source platforms typically may lack a good level of quality as a potential product.

Traditional and typical microcontroller application development process involves C/C++ programming. A source code can be written in C/C++, and the C/C++ compiler will generate a binary file to load it onto the memory area in a microcontroller. Over the last couple of years, Python script language has gained high popularity in higher education and in the open-source community. Particularly, various machine learning (ML) codes and vision processing codes were written in Python, and these useful examples and codes tend to be posted and shared through a collaborative platform such as GitHub. The Python script language also has become an important resource for microcontroller applications due to the fact that microcontroller programs can be written in Python language. For this reason, the Python language can be also taught and potentially adopted in microcontroller/microprocessor application courses.

For resource-rich microprocessor-based systems, a Linux operating system can be used. In this resource-rich environment, a standard Python library can be installed on the system. However, for microcontrollers with a limited computing resource, a standard Python library may not be suitable to be installed on the limited size of the internal memory. For selected microcontrollers, MicroPython can be used instead. MicroPython includes a compact version of a Python programming language interpreter. Raspberry Pi Pico boards are a part of the selected boards supporting MicroPython. In this paper, a Raspberry Pi Pico board with an RP2040 MCU is chosen to show programming examples for selected laboratory examples and example programs. In the following section, the details of the programming environment for the Raspberry Pi Pico board will be presented.

Raspberry Pi and MicroPython

In order to program a Raspberry Pi Pico board, there are several development environments. One of the development environments is to use another Raspberry Pi board such as Raspberry Pi 4 or Raspberry Pi 5 boards. For instance, a Raspberry Pi Pico board can be connected to a Raspberry Pi 4 or 5 system with a keyboard, mouse, and monitor. Using a Python IDE such as Thonny, a user can write a program for the Raspberry Pi Pico relatively easily and fast. Another development environment is to use a PC or Mac. On the PC or Mac, Visual Studio Code can be installed and an additional MicroPython extension may need to be installed [6]. In this way, a Raspberry Pi Pico can be connected and programmed using a PC or Mac. For this development environment, users may need to follow an additional installation process for compilers.

The adopted development environment in this paper is to use a Virtual Machine (VM) such as VirtualBox and install a Raspberry Pi Desktop for PC and Mac on the VM [7]. In this way, a user can experience a Raspberry OS on their laptop, PC, and Mac without the need for a physical unit of a Raspberry Pi 4 or 5 board. After the successful creation of the VM by installing a Raspberry Pi Desktop for PC and Mac. As shown in Figure 1, a Raspberry Pi Pico board can be connected and controlled through the VM.

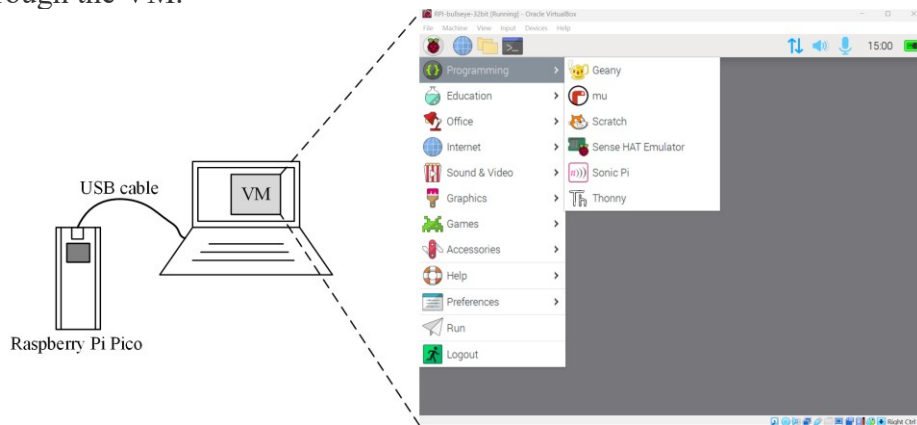


Figure 1. The connection of the Raspberry Pi Pico board to the VM on a laptop/PC.

For the Raspberry Pi OS on the VM, users may need to routinely manage the Linux OS by updating/upgrading the operating system properly. One of the Python IDE that supports the Raspberry Pi Pico board is Thonny. In this paper, Thonny IDE will be used, and simple laboratory examples will be presented in the following sub-section.

Introductory Laboratory Examples using MicroPython

In the author's embedded system course, a BH EDU board is typically used in teaching various electronics and mechanical components that would be controlled by a microcontroller [8]. A BH EDU board is a laboratory board with various components, and it can be used with several microcontroller boards including a Raspberry Pi Pico board. The BH EDU board is not an essential portion of laboratory examples. A reader can create the same set-up by using individual or separate parts instead without using a BH EDU board. A BH EDU board with a Raspberry Pi Pico board is used, and the connection diagram is shown Figure 2 (A). A picture of the BH board with the Raspberry Pi Pico board is shown in Figure 2 (B).

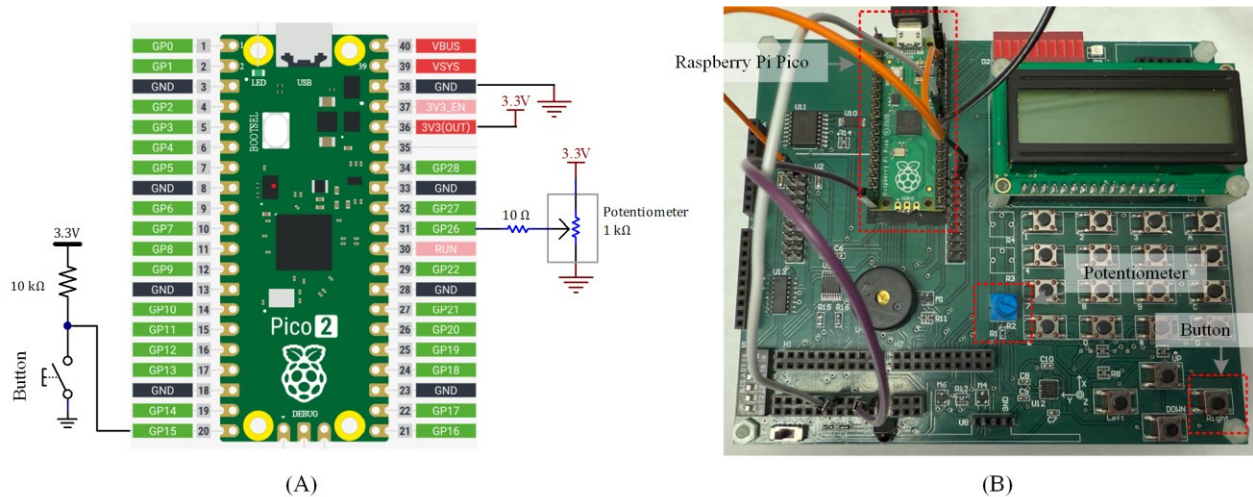


Figure 2. (A) A connection diagram (B) A picture of the Raspberry Pico Board and BH EDU board.

Based on the component connection shown in Figure 2, a MicroPython code was written as a GPIO control example. The code is shown on the left side of Figure 3. For the task described in this program, it is to blink an LED if a button is pressed. The button is connected to GP15. The picture of the LED is shown on the right side of Figure 3. Using this simple GPIO laboratory example, students can learn about reading the status of the input pin, and process the status to control the LED turn on or off.

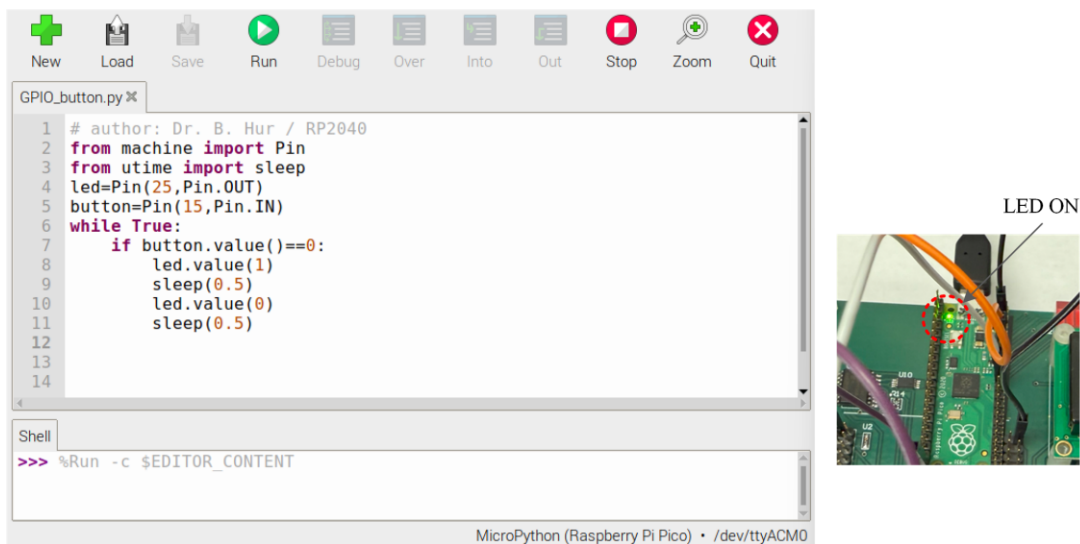
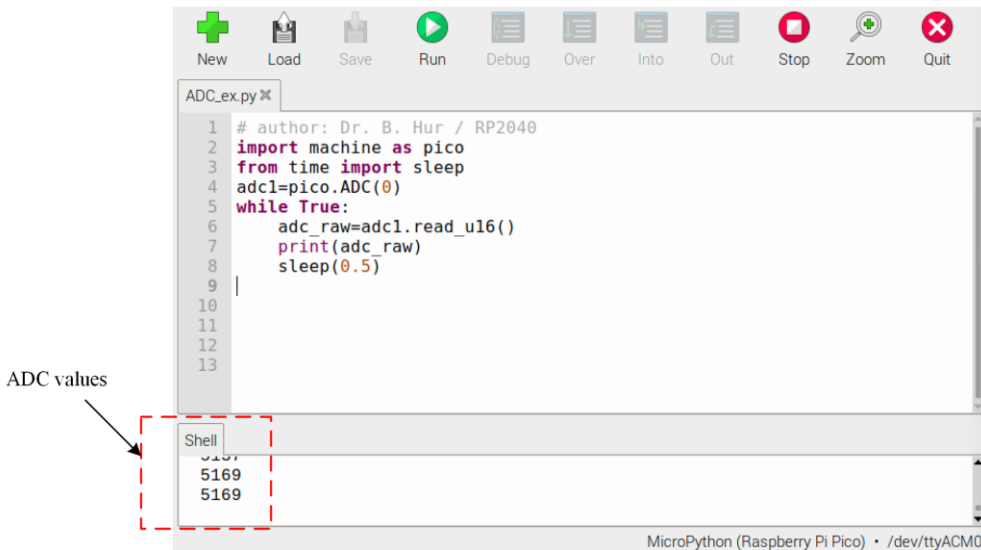


Figure 3. GPIO control example / MicroPython Example 1.

Next, an analog-to-digital converter (ADC) laboratory example will be presented. An ADC module is one of the important and useful peripherals as it can be used for many applications. A single-channel ADC example program is shown in Figure 4. The hardware connection diagram is the same as shown in Figure 2. In this ADC example program, the analog voltage at the GP26 pin will be read and converted to a 16-bit digital value. Then, the converted ADC value will be printed on the shell window as shown in Figure 4. As a user tweaks a potentiometer, the ADC values will be changed. This single-channel ADC example can assist students in learning about the basic usage of the ADC module.

Students may extend their knowledge to multiple channel cases or other ADC applications.



```

1 # author: Dr. B. Hur / RP2040
2 import machine as pico
3 from time import sleep
4 adc1=pico.ADC(0)
5 while True:
6     adc_raw=adc1.read_u16()
7     print(adc_raw)
8     sleep(0.5)
9
10
11
12
13

```

ADC values

Shell

```

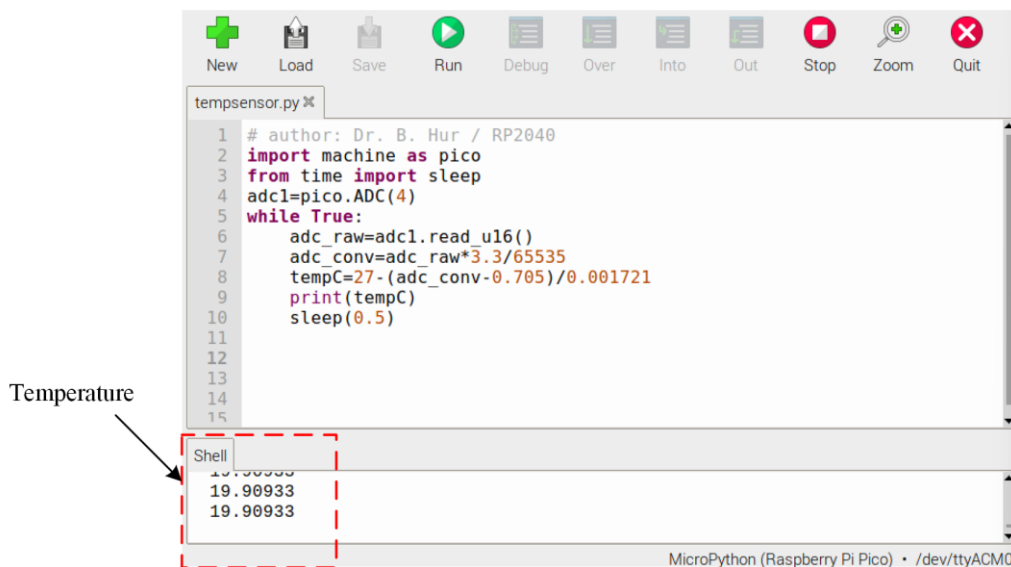
5169
5169

```

MicroPython (Raspberry Pi Pico) • /dev/ttyACM0

Figure 4. A single channel ADC example (GP26) / MiroPython Example 2.

Next, as an ADC application example, a temperature sensor example will be presented. A Raspberry Pi Pico board includes an RP2040 microcontroller. An RP2040 microcontroller includes an internal temperature sensor. The converted digital data for the temperature sensor can be accessed through one of the ADC channels. The temperature sensor example program is shown in Figure 5. This program can read the internal temperature sensor data and convert it into degree Celsius. The converted temperature values will be displayed on the shell window, as shown in Figure 5.



```

1 # author: Dr. B. Hur / RP2040
2 import machine as pico
3 from time import sleep
4 adc1=pico.ADC(4)
5 while True:
6     adc_raw=adc1.read_u16()
7     adc_conv=adc_raw*3.3/65535
8     tempC=27-(adc_conv-0.705)/0.001721
9     print(tempC)
10    sleep(0.5)
11
12
13
14
15

```

Temperature

Shell

```

19.90933
19.90933

```

MicroPython (Raspberry Pi Pico) • /dev/ttyACM0

Figure 5. ADC application (Internal Temperature Sensor) / MiroPython Example 3.

This paper presented three typical simple Micro-Python based laboratory examples for that can be taught and covered in microcontroller application courses. The author created several MicroPython

lesson materials for a microcontroller application course and taught in class. Based on the author's limited experience, the author found many typical C/C++ based laboratories associated with serial communications, motor control, servo control, and so forth could be successfully implemented using Micro-Python programming.

MicroPython programming can manage relatively simple embedded system laboratories quickly and easily. When it comes to implementations of complex functions or behaviors, in some cases, MicroPython programming does not seem a good choice. This is because C/C++ programming can access low-level registers and high-level libraries, and it can be used to control most of the complex behaviors of a microcontroller. Based on the author's limited experience, it seems MicroPython programming can be effectively used in relatively simple functional requirements of a system. In cases where the functional requirements of a system are complex and a developer may need to access certain memory and registers of a microcontroller to implement the complex functions effectively, C/C++ programming can still be a good choice for the implementation of the functions. Further discussion and lessons about MicroPython-based microcontroller education will be continued in the following sub-section.

Discussions and Lessons about MicroPython-based Microcontroller education

Custom microcontroller applications could be built rapidly by using the MicroPython language. Users can meet the required functions with a relatively low effort. As described, for instance, typical functions such as serial communion, motor control, and servo control can be implemented using MicroPython programs. Users who may seek a rapid prototype development environment or non-commercial hobby projects can use boards that support a MicroPython language, and they can complete their assignments and projects effectively and fast. To educators in higher education institutions, Python and MicroPython programming for microcontroller application development can be taught and covered. In the author's opinion, these topics can be covered to provide additional information and tools for students.

However, there can be several problems with Python-based firmware and programs. One of them would be a security problem. The Python/MicroPython code can be relatively easily retrieved from a device. This may become a serious concern for commercial and product-level microcontroller applications. For this reason, in the author's opinion, C/C++-based programming for microcontroller applications would still be the primary choice. Depending on the instructor's choice, Python/MicroPython as well as other open-source tools such as Arduino may be taught in a higher education institution as an additional and/or alternative implementation of microcontroller applications.

The author has taught Python-based programming in his graduate-level advanced embedded system courses over semesters. In his graduate-level course, students learned about C/C++, Assembly, and Python programming. The majority of programming lessons were based on Python. In Spring 2025, the author updated his undergraduate-level embedded systems course materials by the inclusion of the Micro-Python discussed in this paper. The further impact on the students due to the addition of the Micro-Python topics will be monitored and studied.

Summary and Conclusions

In this paper, Python/MicroPython-based microcontroller programs were examined for potential use in higher education institutions. Traditionally, for firmware or microcontroller education, C/C++ languages are used and still, C/C++ programming would be considered as a typical method. In order to study and teach about low-level processor architectures, assembly language can be also covered in microcontroller education. These methods are based on a compiler that can generate executable binary code for microcontrollers and processors. As an alternative method, Python scripts can be used to control microcontrollers and processors. Specifically, MicroPython for Raspberry Pi Pico board was considered for an education purpose. Several typical laboratory examples controlling microcontrollers to perform various functions including GPIO control, ADC, Motor control, Servo control, and serial communications can be implemented using the MicroPython script language. Generally, the Python/MicroPython programs would be useful in cases where a rapid prototype is needed. As it was discussed in this paper, MicroPython-based microcontroller application courses may not necessarily replace the traditional application courses entirely at this point. However, this Python/MicroPython-based programming method is useful for students to learn as additional topics for the microcontroller application course curriculum. The author plans to continue to pursue learning new methods in the microcontroller application area and share the lessons learned through future publications.

References

1. Van Rossum, G., & Drake Jr, F. L. (1995). Python tutorial (Vol. 620). Amsterdam, The Netherlands: Centrum voor Wiskunde en Informatica.
2. Van Rossum, G. (2003). An introduction to Python (p. 115). F. L. Drake (Ed.). Bristol: Network Theory Ltd..
3. MicroPython. [Online]. <https://micropython.org>
4. CircuitPython. [Online]. <https://circuitpython.org>
5. Halfacree, G., & Everard, B. (2021). Get Started with MicroPython on Raspberry Pi Pico: The Official Raspberry Pi Pico Guide. Raspberry Pi Press.
6. Loker, D. (2022, August). Embedded systems using the raspberry pi pico. In 2022 ASEE Annual Conference & Exposition.
7. Raspberry Pi Desktop for PC and Mac. [Online]. <https://www.raspberrypi.com/software/raspberry-pi-desktop>
8. Hur, B. (2022, March). Transition back to in-person class for an embedded system course in Engineering Technology during the COVID-19 pandemic. In 2022 ASEE Gulf Southwest Annual Conference.

BYUL HUR

Dr. B. Hur received his B.S. degree in Electronics Engineering from Yonsei University, in Seoul, Korea, in 2000, and his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Florida, Gainesville, FL, USA, in 2007 and 2011, respectively. In 2016, he joined the faculty of Texas A&M University, College Station, TX, USA, where he is currently an Associate Professor. His research interests include Mixed-signal/RF circuit design and testing, measurement automation, environmental & biomedical data measurement, and educational robotics development.