ASEE 2022 ANNUAL CONFERENCE
Excellence Through Diversity
MINNEAPOLIS, MINNESOTA, JUNE 26TH-29TH, 2022 ASEE
Paper ID #36818

Embedded Systems using the Raspberry Pi Pico

David Loker

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.

Embedded Systems using the Raspberry Pi Pico

Abstract

In engineering and engineering technology programs, students learn about embedded systems in a variety of upper and lower-division courses. Each of these courses includes a lab component where students design and implement hands-on projects which reinforce lecture materials. This paper will demonstrate that the Raspberry Pi Pico embedded board can be used as a basis for many laboratory projects within a variety of embedded courses. Some courses include Introduction to Programming, Microprocessors, Embedded Systems, Object-Oriented Programming, Measurements and Instrumentation, Wireless Communications, and Control Systems. The Raspberry Pi Pico is a new inexpensive board built on the RP2040 microcontroller. Some key features of this board include USB device and host support, 26 GPIO pins, on-board temperature sensor, PWM, ADC, and various interfaces such as SPI, I2C, and UART. Software development can occur using either C or MicroPython. MicroPython is a small subset of the Python standard library, and it is designed to run on a variety of microcontrollers for embedded applications. For this work, MicroPython will be used for software development within Thonny, the Python IDE. The goal of this paper is to show a series of embedded lab projects that can be used within these courses. Possible lab projects include: digital and analog I/O operations, serial LCD interface (I2C and UART), control of LED strip using SPI interface, ultrasonic range sensor control, temperature sensor interface using ADC control, GPS control, and DC motor control using a half-H bridge driver IC. Several of these projects will be presented in this paper. A student assessment will also be provided.

Introduction

In engineering and engineering technology programs, there are many courses that use embedded systems to meet the program requirements. C/C++ programming is often used with embedded hardware and software as a core component to these courses. Examples include the usage of the Programmable System-on-Chip (PSoC 5LP) and the BeagleBone Black (BBB) [1-3]. Alternatively, there is a growing interest in graduates having Python experience. MicroPython is a small subset of the Python standard library, and it is optimized to run on a variety of microcontrollers for embedded applications [4-5].

Embedded Courses

In engineering and engineering technology programs, curriculum is set up so that programming and embedded systems are taught through a variety of courses. These include: Introduction to programming, Digital Design, Introduction to Microprocessors, Object Orientated Programming, Intermediate Embedded Systems, and Advanced Embedded Systems. Additionally, other courses (e.g., Communication Systems, Control Systems, Senior-level Project-based courses, etc.) may include embedded systems as an integral component. After completing these courses, students should be prepared for jobs based on programming and embedded systems.

C/C++ programming is often used with embedded hardware and software as the core component to these courses. An entry level microprocessors course may use an “Arduino” type computer board to teach ‘C’ programming and embedded hardware. Since this is an introductory course,

all of the hardware interfaces are accomplished using API/library calls. Students learn 'C' programming, and with the availability of libraries, they can ignore the low-level register programming.

Intermediate embedded courses provide more depth, where the students can design software that interfaces directly to the hardware. Students reference manufacturer's datasheets and learn how to use 'C' to directly address the microcontroller's control registers. Topics can include interrupts, timers, ADC and DAC interfacing, and filtering. A final lab project can be used to integrate the individual labs developed throughout the semester.

Due to the open-source popularity of Python, some programs are supplementing C/C++ courses with Python. An example is a wireless communications systems course utilizing MicroPython on the Digi XBee3 module [4].

For this paper, an embedded lab project for a 1-credit senior-level seminar course is shown. The primary objective of this course is to help prepare the seniors for the senior design course (offered in the following semester) by covering project management techniques (e.g., requirement specification, design specification, work breakdown structure, gantt chart, etc.). The required embedded lab project for this course was intended to guide the students through the design process for a small project. The project was individual rather than team based. The hardware portion of the project utilized the Raspberry Pi Pico board, and software used MicroPython with Thonny, the Python IDE [6].

Lab Project Requirements

A listing of possible projects was provided to the students. Students were allowed to determine a project of their own choosing.

Some possible examples of projects included:

- Pushbutton Game Controller
- RGB Color Control Using PWM
- GPS Location Finder
- Joystick Control
- Simon Says Game
- Model Railway Road Crossing
- Addressable LED Strip Control
- Greenhouse Temperature Monitor and Control System
- Digital Audio Clip Jukebox
- Traffic Light Control System
- Sound Decibel Meter
- Ultrasonic Range Finder

Students were required to submit a project proposal consisting of the following.

- Project title
- Project introduction (paragraph description of the project and a bullet listing of at least three engineering requirements)

- Architectural block diagram
- Parts list
- References (Students could base part of their project using online resources. However, they need to specify where they obtained this information, and their project must extend beyond the online resources).

Students were also provided with details regarding the project deliverables and grading criteria, and they were required to submit an academic integrity statement.

Project Deliverables

- Executive summary of the results (Word file):
  - Written description (1-2 paragraphs) of the hardware design (also include OrCAD PSpice schematics)
  - Written description (1-2 paragraphs) of the software design (also include a flowchart created in Word, ppt, or other s/w)
  - Testing procedure (numbered step by step testing procedure for each engineering requirement)
  - Results (1-2 paragraphs)
  - Signed academic integrity statement
- 2-3 minute video (posted on YouTube) demonstrating successful completion of the lab project
- Upload to Canvas the following:
  - Word file that contains the executive summary
  - All software source code
  - Link to video

Grading Criteria:

- Executive Summary – 60%
- 2-3 Minute video posted on YouTube – 20%
- Demonstration of successful completion of at least 3 engineering requirements - 20%

MicroPython

Due to the free and open-source software availability of Python and the accessibility of online libraries, there is a growing interest in graduates having Python experience. Some programs are supplementing C/C++ courses with Python. MicroPython is a small subset of the Python standard library, and it is optimized to run on a variety of microcontrollers for embedded applications [7-8].

MicroPython implements functional specific libraries. Some examples include Bluetooth, machine, and network. The machine module contains specific classes of functions related to the hardware on embedded boards. A subset of these classes is listed below.

- class Pin for I/O control pins
- class ADC for analog to digital conversion
- class PWM for pulse width modulation
- class UART for serial communication

- class I2C for a two-wire serial protocol
- class SPI for serial peripheral interface protocol

One device that can be used with MicroPython is the Digi XBee3 module, which implements the Zigbee, IEEE 802.15.4, and BLE protocols [9]. A new inexpensive device is the Raspberry Pi Pico embedded board built on the RP2040 microcontroller that can be programmed using MicroPython.

Raspberry Pi Pico

The Raspberry Pi Pico board has the following features [10-11].

- 23 General Purpose Digital I/O only pins (GP0 – GP22)
- 3 General Purpose pins that can be used for digital I/O (GP26 – GP28) or ADC inputs
- One 12-bit 500ksps ADC
- On-board LED (GP25)
- On-board temp sensor
- Digital peripherals:
    - 2 UART (UART0/1 TX, RX)
    - 2 I2C (I2C0/1 SDA, SCL)
    - 2 SPI (SPI0/1 SCK, TX, RX, CSn)
    - 16 PWM (General Purpose pins – 8 slices, each with two outputs)

Pinouts for the Raspberry Pi Pico board are shown in Figure 1.

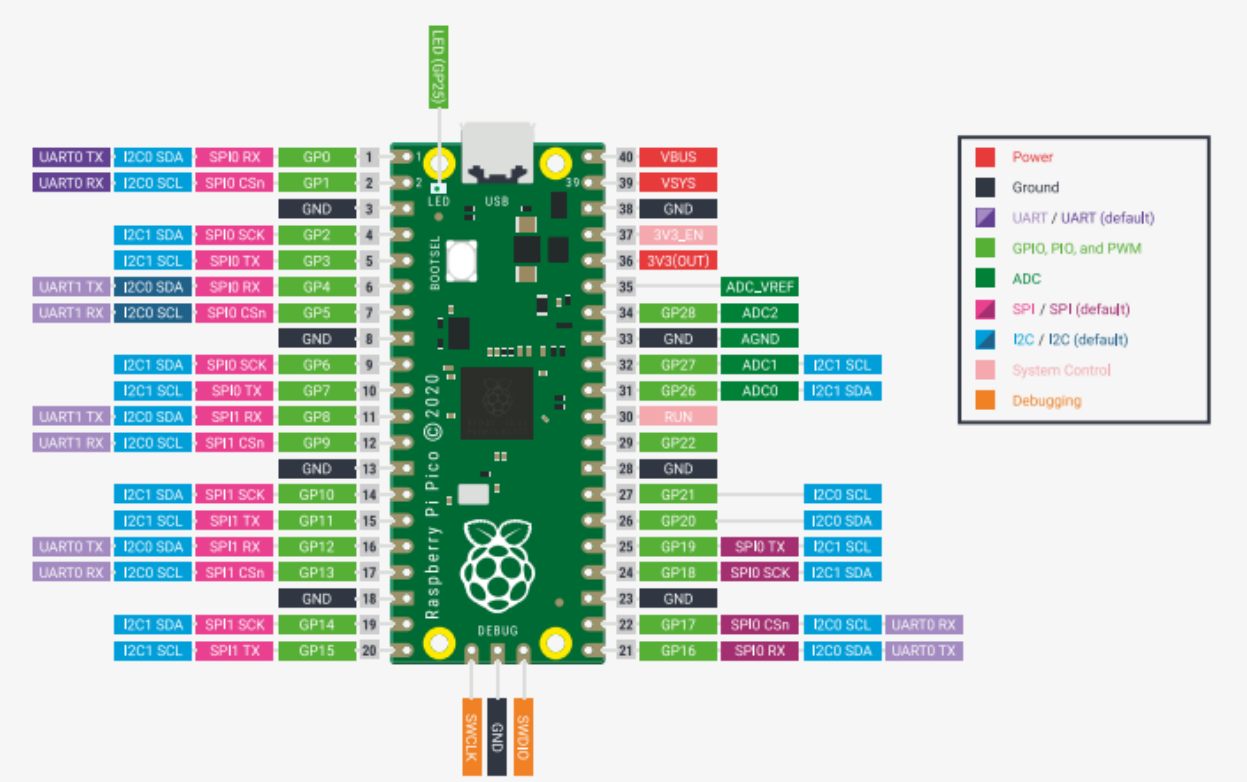

Figure 1. Pico Pinout (shown on pg. 4 of [10])

MicroPython Code

Snippets of MicroPython code from various students' projects are shown in Figure 2.

- For the analog input, the ADC class with the read_u16() method returns an integer value between 0 and (2^{16}-1). An LM35 with a temperature coefficient of 10mV/ºC can be connected to one of the ADC inputs. The ADC has a full-scale voltage of 3.3V and 12 bits of resolution. Equations for temperature conversion are shown in the code.
- For digital I/O, the Pin.IN and Pin.OUT constants (within the Pin class) are used to configure the pin objects. Devices are connected to the digital GPIO pins specified. Pushbuttons can be used for inputs, and LEDs and buzzers can be used for outputs.
- For the I2C LCD, i2c_lcd.py and lcd_api.py are libraries available for download [12]. Parameters for both the I2C and LCD are specified. Then, various methods (e.g., clear(), move_to(), and putstr()) are used for clearing, moving the cursor, and placing text on the LCD.
- For PWM control, the PWM class is used for specifying output pins. Also, freq() and duty_u16() methods are used for controlling the PWM output frequency and duty cycle.
- The UART class implements the serial communications protocol. The any() method returns the number of characters available to read, and the read() method reads the characters.

Thonny IDE

An IDE for developing Python code and downloading it to the Pico is called Thonny. It has a script area for writing code and a Python Shell for providing information about running programs. An example is shown in Figure 3. This program uses the on-board temperature sensor (base-emitter junction of a biased bipolar diode) connected to ADC4. The ADC has a full-scale voltage of 3.3V and has 12 bits of resolution. However, it is read using 2 bytes. From the datasheets, Vbe = 0.706V at 27 degrees C, with a slope of -1.721mV/degree. Thus, the temperature can be defined in the Python code. The program prints the temperature to the console, waits for 1 second, and then repeats.

```
1.  from machine import Pin, ADC, PWM, UART  #Imports from the machine library
2.  from lcd_api import LcdApi               #Import the LCD API
3.  from i2c_lcd import I2cLcd               #Import the I2C LCD
4.
5.  #ADC code
6.  #Information for the LM35 calculations
7.  AV = ADC(28)               #Obtains an analog value from GPIO 28
8.  convert = 3.3/65535        #Determine the resolution
9.  raw_volt = AV.read_u16()   #Raw value from Analog Pin (GPIO 28)
10. convert_volt = raw_volt * convert #Take raw value and convert to voltage
11. C_Temp = (convert_volt/(10/1000)) #Convert to Temperature in Celsius
12.
13. #Digital Inputs and Outputs
14. RED = Pin(2, Pin.OUT)      #Red pin of RGB on pin 2
15. GREEN = Pin(3, Pin.OUT)    #Green pin of RGB on pin 3
16. BLUE = Pin(4, Pin.OUT)     #Blue pin of RGB on pin 4
17. PB_MANUAL = Pin(5, Pin.IN) #Push button used for manual mode set on pin 5
18. SW_DEG_C = Pin(6, Pin.IN)  #Switch for degree C on pin 6
19. SW_DEG_F = Pin(7, Pin.IN)  #switch for degree F on pin 7
20. BUZZER = Pin(10, Pin.OUT)  #buzzer for manual mode on pin 10
21.
22. #Information for the I2C LCD
23. I2C_ADDR = 0x27
24. #OR I2C_ADDR = i2c.scan()[0]
25. I2C_NUM_ROWS = 2     #There are two rows with the I2C LCD
26. I2C_NUM_COLUMNS = 16 #There are 16 columns with the I2C LCD
27. i2c = I2C(0, sda = Pin(0), scl = Pin(1), freq = 400000) #sets up I2C parameters
28. lcd = I2cLcd(i2c, I2C_ADDR, I2C_NUM_ROWS, I2C_NUM_COLUMNS)
29. lcd.clear()          #Clears the LCD
30. lcd.move_to(0,0)     #Moves the start of the LCD text to position 0,0
31. #Prints the temp in F in the format XX.X
32. lcd.putstr("Manual   " + "       Temp in F: " + str(round(F_Temp, 1)))
33.
34. #PWM control
35. #Three PWM pins to control the RGB LED
36. Red = PWM(Pin(20))
37. Green = PWM(Pin(19))
38. Blue = PWM(Pin(18))
39. #Sets the PWM freq. for the RGB
40. Red.freq(1000)
41. Green.freq(1000)
42. Blue.freq(1000)
43. #Sets an initial duty cycle of 0 for each color. Max d.c. = 65535
44. Red.duty_u16(0)
45. Green.duty_u16(0)
46. Blue.duty_u16(0)
47.
48. #GPS Module UART Connection
49. gps_module = UART(0, baudrate=9600, tx=Pin(16), rx=Pin(17))
50. length = gps_module.any()
51. gps_module.read(length)
52.
```

Figure 2 – MicroPython Code Snippets

Pico_temp_sensor1.py

```
1  from machine import ADC
2  from time import sleep
3  sensor_temp = ADC(4)
4  conversion_factor = 3.3 / (65535)
5  while True:
6      reading = sensor_temp.read_u16() * conversion_factor
7      temperature = 27 - (reading - 0.706)/0.001721
8      print(temperature)
9      sleep(1)
```

Shell

```
MicroPython v1.17 on 2021-09-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  18.61781
  18.61781
  18.61781
  17.68152
  18.61781
  19.55409
  20.02224
  20.02224
  20.49038
  20.49038
  20.49038
  20.49038
  20.49038
  20.95853
  21.42667
  20.49038
  20.95853
  20.95853
  20.95853

MicroPython v1.17 on 2021-09-02; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

Figure 3 – Pico Temp Sensor Code

Student Project Examples

<u>GPS Location Finder Project</u>

For this project, several engineering requirements were as follows:

- Display settings controlled by a switch
    - User's latitude and longitude coordinates, and the user's heading.
    - Date, time, and user's speed
- System must be portable.
- System updates every second
- System components:

- Raspberry Pi Pico
- GT-U7 GPS Module [13]
- 16x2 (1602) LCD Display Module [14]

The hardware components for the project and the results displayed on the LCD are shown in Figures 4 and 5.

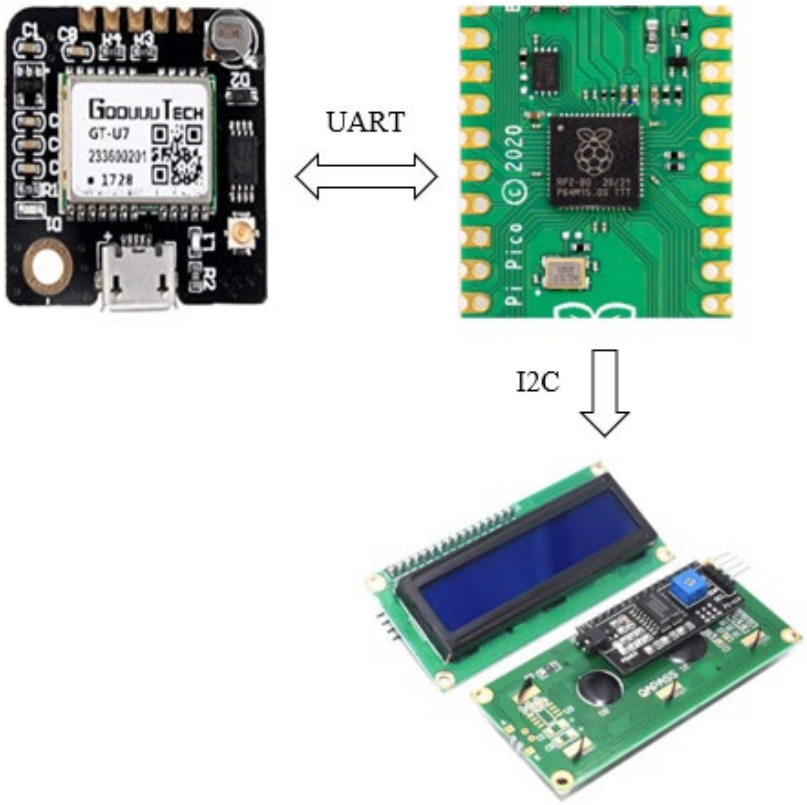

Figure 4. Hardware Components for GPS Project

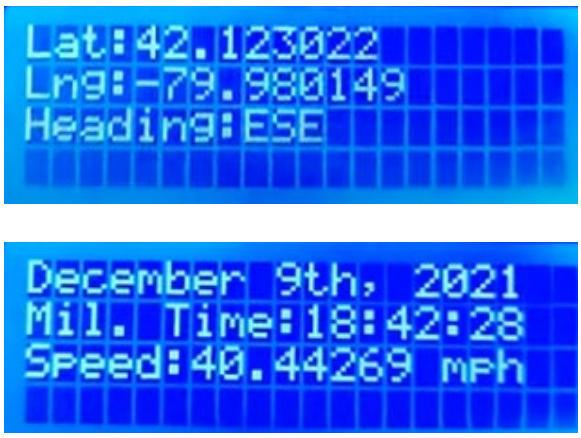

Figure 5. LCD Display Results for GPS Project

The software application used the following classes.

- I2C for displaying text on the LCD
- Pin for specifying the pin connections to the switch, I2C SDA and SCL pins, and UART TX and RX pins
- UART for specifying the baudrate

Simon Says Game Project

The overall objective is to have the user select the appropriate sequence of colored pushbuttons that matches the sequence of colors displayed on an RGB LED. The engineering requirements for the "simon says" project are:

- Game uses three possible levels of difficulty
    - Easy – 3 rounds
    - Medium – 4 rounds
    - Hard – 5 rounds
- System uses an LCD to communicates results with the user
    - Shows the level of difficulty selected
    - Specifies the round number
    - Shows whether you win or lose
- Colors implemented
    - Red
    - Green
    - Blue
    - White
    - Orange
    - Yellow

The hardware components for this project consisted of:

- Raspberry Pi Pico
- One RGB LED
- Three pushbuttons for selecting the level of difficulty
- Six colored pushbuttons for selecting the sequence of colors
- 16x2 (1602) LCD Display Module

The software application used the random library for generating a random sequence of colors. It also used the following classes.

- I2C for displaying text on the LCD
- Pin for specifying the pin connections to the pushbuttons and LED
- PWM for specifying the duty cycle for each of the three pins that control the colors for the RGB LED

Ultrasonic Range Finder Project

The overall objective of this project is to measure the distance to an object. The engineering requirements consist of:

- Determine the distance to an object in cm.

- Display distance on an LCD
- Control the color of an RGB LED based on the distance to the object

The hardware components for this project consisted of:

- Raspberry Pi Pico
- One RGB LED
- 16x2 (1602) LCD Display Module
- HC-SR04 Ultrasonic Ranging Module (See Figure 6) [15]

The software application used the ticks_us() function (within the utime library) for determining the pulsewidth in microseconds of the returned echo for calculating distance. It also used the following classes.

- I2C for displaying distance on the LCD
- Pin for specifying the pin connections to the RGB LED, I2C SDA and SCL pins, and trigger and echo pins for the HC-SR04
- PWM for specifying the duty cycle for each of the three pins that control the colors for the RGB LED

Figure 6. HC-SR04 Ultrasonic Ranging Module

Student Assessment

Table 1 provides results from a student questionnaire regarding the lab project. This was an online questionnaire completed at the end of the semester, and students were given some extra credit points for homework. A total of 13 students completed the questionnaire. Most of the students spent 11-20 hours working on their project. A significant majority of the students found the overall level of difficulty for using the Pico and Thonny to be easy, online resources were easy to find, and level of interest in using the Pico on future projects to be positive. Additionally, 100% of the students indicated that the Pico should be used in the ECET program. Nearly all of the students provided additional positive comments regarding the Pico.

Table 1. Student Questionnaire Results.

Questions	Responses
1. Approximate time in hours you worked on your project • 1-10 • 11-20 • 21-30 • 31-40 • > 40	• 1-10 (23%) • 11-20 (46%) • 21-30 (31%) • 31-40 (0%) • > 40 (0%)
2. Level of difficulty for using the Pico (e.g., easy, moderately difficult, extremely difficult)	• Easy (62%) • Moderately difficult (38%) • Extremely difficult (0%)
3. Level of difficulty for using Thonny (e.g., easy, moderately difficult, extremely difficult)	• Easy (85%) • Moderately difficult (15%) • Extremely difficult (0%)
4. Availability of online resources for using the Pico (e.g., easy to find, moderately difficult to find, extremely difficult to find)	• Easy (85%) • Moderately difficult (15%) • Extremely difficult (0%)
5. Should the Pico be taught within the ECET program? If so, where?	• Yes (100%) • No suggestions specified.
6. What is your level of interest in using the Pico for future projects? (e.g., positive, neutral, negative)	• Positive (69%) • Neutral (31%) • Negative (0%)
7. Please provide any additional comments.	• Comments are listed separately.

A listing of the additional comments provided by the students is below.

- Programming the Pico using MicroPython is relatively straight forward and is definitely worth being used in future classes.
- I enjoyed working with Pico. I was able to research on python programming.
- I had a lot of fun learning to program with the Pico.
- The pico was very easy to just pick up and start coding with right away. There are a lot of resources online to help out if you run into a problem. The board was also extremely nice to use as almost every port was able to be configured for multiple things such as PWM.
- Versus the PIC18 series of microcontrollers, the Pico is significantly more recent, user-friendly, and powerful. A lot of really basic things within the PIC "ecosystem" are needlessly complex given how often they're used. Ex: Setting up a timer/wait/clock output is at least a

dozen lines of assembly and maybe half that in C. Python is something to the effect of importing the time library and calling the sleep() function. That being said, in my project I did find myself limited by Python. I think using C with the Pico should be explored as an option as well. Really both languages on the Pico aren't that different due to the dozens of libraries supported by both, but some things are just easier with Python vs. C and vice versa.

- I liked the project and the need to use Micropython. Python in general will be an important tool to have in the future and I can see a need to include the pico or similar python projects for this major in other courses.
- It seems the Arduino IDE is used more (thus more libraries) than Thonny, but I guess it just boils down to if you prefer the C++ or Python.
- Overall a fun project.
- I believe the Pico project was a decent addition to the course structure. I really enjoyed working with the pico.
- Its definitely a good idea to implement into the ECET program. It gives more experience in programming microcontrollers and how to integrate them into circuits. The software that can be used can program in several different languages and the hardware after the class is actually useful for other applications. It is also beneficial to have students working with the most modern hardware possible to give them an edge up when trying to find a job for after their schooling.

Summary

The Raspberry Pi Pico is an inexpensive board suitable for embedded applications in a variety of courses. MicroPython is optimized to run on a variety of microcontrollers, including the Pico board. Thonny, the Python IDE, is available for free and can be used as the software development environment.

The lab projects presented in this paper are only a subset of the various projects that can be implemented using the Pico board. A large majority of the students found the overall level of difficulty for using the Pico and Thonny to be easy, online resources were readily available, and level of interest in using the Pico on future projects to be positive. Additionally, 100% of the students indicated that the Pico should be used in the ECET program.

Students successfully demonstrated various embedded programming applications using MicroPython with Thonny on the Pico board. With the ease of programming using MicroPython, this provided an effective addition to embedded C/C++ programming applications. Overall, the inexpensive Pico board and MicroPython can be used for many embedded applications in a variety of courses.

References

[1] S. Strom and D. Loker, "Programmable System-On-Chip (PSoC) Usage in an Engineering Technology Program," *Annual Meeting, American Society for Engineering Education,* 2016.
[2] D. Loker and S. Strom, "Programmable System-On-Chip (PSoC) Usage in Embedded Programming Courses," *Annual Meeting, American Society for Engineering Education,* 2020.
[3] S. Strom and D. Loker, "BeagleBone Black for Embedded Measurement and Control Applications," *Annual Meeting, American Society for Engineering Education,* 2018.
[4] D. Loker, "MicroPython in a Wireless Communications Systems Course," *Annual Meeting, American Society for Engineering Education,* 2021.
[6] thonny.org. [Online]. Available: https://thonny.org/
[7] MicroPython.org. [Online]. Available: https://docs.micropython.org/en/latest/
[8] MicroPython.org. [Online]. Available: http://www.mircropython.org
[9] Digi.com. [Online]. Available: https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee3-zigbee-3
[10] Raspberrypi.com. [Online]. Available: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf
[11] G. Halfacree and B. Everard, *Get Started with MicroPython on Rapsberry Pi Pico*, Raspberry Pi Trading Ltd, 2021.
[12] github.com. [Online]. Available: https://github.com/dhylands/python_lcd/tree/master/lcd
[13] Amazon.com: [Online]. Available: https://www.amazon.com/Navigation-Satellite-Compatible-Microcontroller-Geekstory/dp/B07PRGBLX7
[14] handsontec.com. [Online]. Available: http://www.handsontec.com/dataspecs/module/I2C_1602_LCD.pdf
[15] SparkFun.com. [Online]. Available: https://www.sparkfun.com/products/15569