The ESP32 microcontroller as a low-cost teaching tool for mechatronics

Abrie J Oberholster
Department of Mechanical and Aeronautical Engineering, University of Pretoria, South Africa, abrie.oberholster@up.ac.za

Abstract

South African universities face unique challenges in teaching technology-intensive modules such as mechatronics. One such challenge is the variability in students' prior exposure to technology and programming. As such, not all students are equally confident in the technological skills required for the final year mechatronics and control module offered by the Department of Mechanical and Aeronautical Engineering at the University of Pretoria.

However, recent technological advancements have opened up new possibilities. The ESP32-WROOM microcontroller, in particular, offers a versatile platform for learning mechatronics. Its compatibility with multiple programming languages ensures that students can choose the language they are most comfortable with to explore the world of mechatronics. The ESP32-WROOM's built-in wireless capabilities further make it ideal for Internet of Things (IoT) applications.

This paper focuses on the selection of microcontrollers and their utilisation in the mechatronics course offered by the university.

Keywords: Microcontrollers, Mechatronics, Programming

1 Introduction

South African universities face unique challenges in teaching technology-intensive modules such as mechatronics. A major contributing factor is that students do not have equal prior exposure to technology and programming at the primary and secondary school levels (Faloye & Faniran, 2023). One of the drivers of this “digital divide” is inadequate information and communication technology (ICT) infrastructure at schools due to socio-economic factors, with learners from affected schools reported to constitute 66% of the total public schooling system in South Africa (Chisango & Marongwe, 2021; Sithomola, 2021). As such, these students are not sufficiently equipped with basic computer and data manipulation skills at the school level. Furthermore, most strategies universities adopt to accommodate these students involve a ‘one-size-fits-all’ approach, which often does not address the challenges of individual students (Faloye & Faniran, 2023). Subsequently, not all students are equally comfortable with the programming skills required for an undergraduate course in mechatronics.

One solution is block-based visual programming (BBVP) which, as a programming tool, has gained wide popularity globally and can be an effective environment for introducing learners with minimal ICT skills to programming (Stolpe & Hallström, 2023; Weintrop, 2019). What makes high-level BBVP tools (such as the Scratch software) powerful in this scenario is that they bypass the requirement for good typing skills and make it possible to use natural language in the description of the behaviour of a command. Several BBVP tools are available for free, making BBVP further accessible. BBVP is used to teach programming skills, ranging from robotics for children (Stolpe & Hallström, 2023) to industrial robotics for adult novices (Weintrop, 2019) to programming for university students (Harvard University, n.d.). For advanced

applications, MATLAB Simulink and National Instruments LabVIEW represent commercial BBVP tools typically used in industry and academic research for system and process control.

It is essential to recognise that BBVP utilises formal programming languages such as Python and C++, and acts as an intermediary between the user and these traditional languages. For advanced applications however, the user is required to be familiar with one or more of these formal programming languages.

Python and C++ are among the most popular programming languages available. While Python is considered a high-level programming language, C++ is classified as a mid-level language. Python is generally less complex than C++, but much slower than C++ for specific tasks. Since Python code does not require compilation as with C++, Python's readability and ease of use make it a valuable script prototyping and lecturing tool in mechatronics. Higher-speed computations are necessary for more advanced applications, such as controlling an inverted pendulum. In such instances, C++ outperforms Python (Balogun MO, 2022; Capaciteam, n.d.).

In 2022, the opportunity arose to transform the classical control systems module offered by the University of Pretoria into a more applied mechatronics and control course. The challenge of selecting an appropriate microcontroller for incorporation into the module was recognised at the beginning of this endeavour. The requirement to balance the need to accommodate students with varying programming proficiency levels while teaching more advanced mechatronics concepts was also identified. Additionally, it was important to consider long-term support, local availability, cost efficiency and the presence of an established user community. While the previous version of the control systems module utilised powerful microcontrollers, these became obsolete over time, and did not meet several of the abovementioned requirements. As such, it was necessary to identify a new suitable microcontroller to maximise the educational potential of the module.

2 Microcontroller selection

In line with the observations made in the previous section, it was important to take cognisance of available software options that interface with microcontrollers, to inform the choice of microcontroller. An internet search quickly reveals that there are several software options available. Conducting an extensive survey of all available software options, however, lies outside the scope of this contribution. Instead, the focus was on determining which software options were available for interfacing with the different microcontrollers under evaluation, while addressing a spectrum of needs stretching from basic introductory mechatronics to advanced mechatronics. Examples of suitable software options identified included Blockly (BBVP), Thonny IDE (Micropython), Arduino IDE (C++) and MATLAB Simulink.

Subsequently, the following microcontrollers were evaluated: Arduino Uno, ESP32-WROOM, M5Stack Core 2 and Raspberry Pi 5. Although many more microcontroller options are available, extending the evaluation to more microcontrollers was not possible due to time constraints in selecting a microcontroller for the 2024 rendition of the mechatronics module.

Arduino Uno and Raspberry Pi were considered, as these are legacy devices with excellent support and extensive user communities. At the same time, M5Stack presents a "plug-and-play" option, albeit with limited local availability. The ESP32 microcontroller utilised by M5Stack showed much potential; hence, the ESP32-WROOM microcontroller was also considered. The advantage of this approach is that the microcontroller can be bought separately, reducing initial costs and making it user-scalable.

In evaluating the different microcontrollers, cost-efficiency was considered, taking into account additional costs required for the Arduino Uno, ESP32-WROOM and Raspberry Pi to include sensors, etc., as part of a microcontroller kit to maximise educational value. This was done to compare these with the M5Stack Core2, which has several onboard sensors.

For each of the four microcontrollers, consideration was given to whether they can be programmed via BBVP, Python and C++. Additionally, consideration was given to whether MATLAB Simulink supports the said controller without requiring additional third-party software licenses.

A 6-point Likert scale was used to assign criterion weight, ranging from 0 (No Importance) to 5 (Critical importance). Likewise, a 4-point Likert scale was used to evaluate microcontroller compliance with the set criteria, ranging from 0 (No Compliance) to 3 (Full Compliance).

Table 1 summarises the evaluation of the identified microcontrollers, along with the criteria considered, the assigned weights and scores.

Table 1: Comparison of candidate microcontrollers

Criterion	Weight	Arduino Uno	ESP32-WROOM	M5Stack Core2	Raspberry Pi 5
Cost efficiency	5	3	2	2	1
Local availability	5	3	3	2	2
Programming language	5	2	3	3	3
MATLAB Simulink support	5	3	2	2	3
Processing power	5	1	2	2	3
Onboard connectivity	3	0	3	3	3
User community size	5	3	3	1	3
Low complexity	4	3	2	2	1
Score		87	97	77	88

From the table, it is observed that the ESP32-WROOM achieved the highest score for this particular application. Hence, it was selected for implementation in the specific mechatronics module.

3 Demonstration board development

Since no demonstration boards were available for the ESP32-WROOM at the time, a local company was approached to develop a customised board according to the needs of the university. The resulting board consists of a rotary potentiometer, 6-axis accelerometer, light-dependent resistor (LDR), temperature, humidity and barometric pressure sensor, IR receiver, 0.96" LED screen, RGB LED, red and blue LEDs, two switches, buzzer and an SD card slot. This allows classroom demonstrations and teaching on several digital communication protocols (including I^2C and SPI), pulse width modulation (PWM), analogue-to-digital conversion (ADC), digital-to-analogue conversion (DAC), sensor selection, aliasing, hardware and software interrupts, etc. The final product is shown in Figure 1, which is commercially available on the company's website with the ESP32-WROOM microcontroller for ZAR 895 (at the time of paper submission). Table 2 lists the different components on the board.

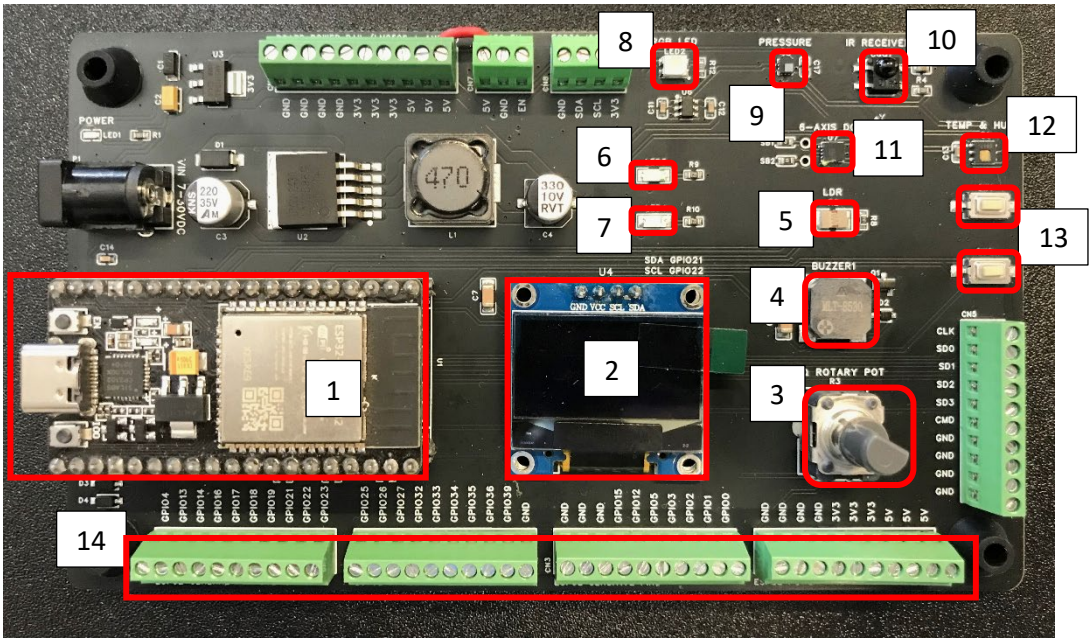

Figure 1: ESP32 Easy Proto V2 (Microrobotics, 2025)

Table 2: ESP32 Easy Proto V2 components

Item	Description
1	ESP32-WROOM (Dev Kit C)
2	0.96” LED screen
3	Rotary potentiometer
4	Buzzer
5	Light-dependent resistor
6	Red LED
7	Blue LED
8	RGB LED
9	Barometric pressure and temperature sensor
10	IR receiver
11	6-axis accelerometer
12	Temperature and humidity sensor
13	Push-button switches
14	Screw terminals

4 Implementation of the ESP32-WROOM

4.1 Practicalities around incorporation into the module

During the 2024 rendition of the mechatronics module presented by the University of Pretoria, the ESP32 Easy Proto V2 was introduced. With the onboard screw terminals, capacitating the measurement of sensor and communication signals with an oscilloscope, this made the unit ideal for teaching and demonstrating digital signal communication protocols and signal sampling theory. Thonny IDE was beneficial for class demonstrations with the board, as no compilation is required, saving time. In a practical session, students

had the opportunity to utilise the ESP32 Easy Proto V2, where they programmed the ESP32 to adjust the RGB LED colour according to the board's tilt using the onboard 6-axis accelerometer as the sensor.

Several final-year students were further tasked with designing and building a rotary inverted pendulum (RIP) using an ESP32 microcontroller as part of their final-year project. An important outcome was that although Thonny IDE was appropriate as a teaching tool, it was too slow to be used in higher-demand applications such as controlling RIPs. C++ was thus identified for higher-demand applications. While Arduino IDE presents an option for C++ programming, MATLAB Simulink combines the power of BBVP with C++. Although MATLAB Simulink does not yet have dedicated hardware support for ESP32, one can interface with ESP32 via the Arduino hardware support package.

While integrating MATLAB Simulink with the ESP32 necessitates a compilation step, it holds the advantage that the user can change parameters during runtime. This is particularly useful when doing live demonstrations in class of the effects of constant values in a PID control loop.

The widespread availability of low-cost, off-the-shelf electronics further enables students to experiment with mechatronics and control systems at home with a kit consisting of an ESP32 microcontroller, H-bridge motor controller and small DC motor with Hall encoder, from as little as ZAR 270 from local resellers.

4.2 Observations on learning efficiency

Students are given the opportunity to give anonymous feedback after each lecture of the mechatronics module, via a Google Form. Prior to incorporating the ESP32-WROOM microcontroller into the module, feedback was often received from students struggling to understand how different aspects of the module connected. This was further exacerbated by the requirement to use C++ (unfamiliar to the students) for low-level programming of the previous microcontroller used in the module.

With the ESP32 microcontroller, MATLAB Simulink was utilised to demonstrate the basic working principles of a quadrature rotary encoder, showing the analogue output signals of the encoder, and how these are used to measure the angle and direction of rotation. It further demonstrated the effects of sample frequency on the measured signal (such as aliasing) and how analogue voltage signals are converted to binary signals. This simple demonstration exposed students to a typical sensor, its operating principle, and basics around signal sampling theory, analogue-to-digital conversion and signal processing. Using MATLAB Simulink helped keep students focused on the learning outcomes of the demonstration, while minimising distractions from coding complexities.

Using the ESP32 Easy Proto V2 with a PicoScope USB oscilloscope, digital communication protocols could be demonstrated effectively in class. Students were observed to be much more engaged with the topic than when they were only shown static textbook figures. Of particular value was the demonstration of the IR receiver, where students became aware that they use a digital communication protocol every time they use an IR remote control.

Using the Thonny IDE as a Python programming interface for the ESP32 Easy Proto V2 helped reduce distractions from low-level coding complexities, which had been problematic in the past. This shift allowed for more in-depth student engagement with topics such as interrupt service routines, analogue-to-digital conversion, bitwise operations, and binary concatenation within the context of the ESP32-WROOM microcontroller.

5 Conclusions

The availability of various software options and low-cost, off-the-shelf electronic hardware, such as the ESP32-WROOM, creates an ideal situation for teaching technology-intensive modules such as mechatronics in an environment where students have a wide range of programming skills. Classroom observations show increased student engagement, allowing them to interact with mechatronics principles hands-on.

This also enables students to experiment with mechatronics in their own capacity at home, and at minimal cost. This teaching and learning approach ultimately aims to empower students to identify and solve everyday mechatronics-related problems innovatively, utilising the available technologies.

6 References

Balogun MO. (2022). Comparative Analysis of Complexity of C++ and Python Programming Languages. *Asian Journal of Social Science and Management Technology*, *4*(2), 2313–7410. www.ajssmt.com

Capaciteam. (n.d.). *C++ vs. Python: All You Need to Know*. Retrieved February 27, 2025, from https://capaciteam.com/c-plus-plus-vs-python/

Chisango, G., & Marongwe, N. (2021). The digital divide at three disadvantaged secondary schools in Gauteng, South Africa. *Journal of Education (South Africa)*, *82*, 149–165. https://doi.org/10.17159/2520-9868/i82a09

Faloye, S. T., & Faniran, V. (2023). Integrating technology in teaching and learning practices: students' competencies. *South African Computer Journal*, *35*(1), 101–114. https://doi.org/10.18489/sacj.v35i1.1111

Harvard University. (n.d.). *CS50's Introduction to Programming with Scratch*. Retrieved February 24, 2025, from https://pll.harvard.edu/course/cs50s-introduction-programming-scratch

Microrobotics. (2025). *ESP32 Easy Proto*. https://www.robotics.org.za/ESP32-EASY-PROTO-V2

Sithomola, T. (2021). The Manifestation of Dual Socio-Economic Strata Within the South African Schooling System A Setback for Congruous Prospects of 4IR. In *African Journal of Public Affairs* (Vol. 12).

Stolpe, K., & Hallström, J. (2023). Visual Programming as a Tool for Developing Knowledge in STEM Subjects. In *Programming and Computational Thinking in Technology Education* (pp. 130–169). BRILL. https://doi.org/10.1163/9789004687912_007

Weintrop, D. (2019). Education block-based programming in computer science education. *Communications of the ACM*, *62*(8), 22–25. https://doi.org/10.1145/3341221