A Block–based IDE Extension for the ESP32

Patrick Lamprecht, Simon Haller-Seeber and Justus Piater

Department of Computer Science, University of Innsbruck
https://informatik.uibk.ac.at/
Technikerstr. 21a, 6020 Innsbruck, Austria

Abstract. Robotics with state-of-the-art microcontrollers leads to plenty of unique opportunities for computer science education at school. This paper introduces an ESP32 extension to Ardublockly specifically designed for educational purposes. It discusses the advantages of block-based programming for school education and presents the new key features within the IDE. To demonstrate the capabilities of the developed extension for computer science education in schools, an exercise in the field of swarm robotics was developed.

Keywords: physical computing, swarm robotics at school, block-based programming

1 Introduction

When computer science teachers want to address new content such as physical computing by building real interactive systems that sense and respond within the real world, they still might use Arduino microcontrollers. Almost all researched sources use or recommend Arduino Uno boards as hardware to implement their (physical computing) projects [5,9,10,11]. The most recent revision 3 of the Arduino Uno came onto the market in 2010 and was not further developed [4, p.5]. More modern microcontrollers such as the BBC Micro:Bit started to emerge in 2015 and were spread with huge efforts by BBC and other partners. The Micro:Bit Educational Foundation (established in September 2016) pushed this local educational experiment to a global scale. One of their big contribution was the vision to develop an "inexpensive, powerful and easy-to-use learning tool" [1, p. 1]. Although one of the core design goals of the Micro:Bit is to "open a window into the future" [1, p. 1–2] this failed due to only providing Bluetooth connectivity and no WiFi on the chip, which makes it hard to cover the evolving field of IoT applications [8].

The ESP32 from Espressif Systems entered the market in September 2016 as a powerful, low-cost, low-power microcontroller. A device with 30 times more processor power and 260 times more SRAM compared to an Arduino Uno Revision 3, as well as a small form factor, low weight, embedded WiFi and Bluetooth, the ESP32 has been evaluated as an excellent microcontroller for IoT scenarios. The bread board friendly version ESP32-DevKitC is treated as an alternative solution for educational purposes. [6, p. 8–9] [7, p.143]

As each pupil owns an individual set of previous knowledge and skills in computer science it is our responsibility as teachers to manage these heterogeneous knowledge levels and to support the learning of pupils in all these different (class) levels. This creates the didactic demand to provide a flexible, block-based Visual Programming IDE to effectively teach programming skills within a physical computing environment.

To address those challenges and to introduce the microcontroller ESP32 to computer science education in schools, we developed an extension to Ardublockly. The extension offers ESP32 support and IoT key features, and is developed to boost the use of robotics in schools. A case study at the end of this paper will demonstrate how this block-based IDE can be used to develop swarm-robotics scenarios with learners in schools. This field was chosen to provide a beneficial new direction for robotics in education, as swarm robotics is based on interactive communication between robots and is very rarely discussed in the field of computer science education.

2 Block-based programming in education

Typically *programming* is perceived as the act of writing program code in a problem-oriented programming language. Once we set the focus to the entire creation process of a program it turns out that the process of developing an algorithm – as a strategy to solve the problem – is a much more demanding task than merely translating it into a programming language. Fuchs introduced the term *Algorithmic Thinking* as the brain activity which is needed to identify algorithmic structures (like branches, loops and conditions) to solve a specific problem. [3, p. 218]

Block-based programming follows this concept of *Algorithmic Thinking* as learners do not need to have skills in any program language syntax. They rather need to show their problem-solving competence as they recognize and apply control and data structures correctly to develop an executable program solving particular problems. [6, p. 3]

The Cognitive Load Theory of Sweller [2] revealed that the overload of a learner's brain has negative impact on learning results. Block-based programming delivers a positive effect on learning, as it sets a strong focus on creating the algorithm, while at the same time taking the burden from learners to code syntax in a text-based programming language. Wintrop and Wilensky [13] confirmed in their field study that it is much easier for pupils to learn block-based programming than text-based programming. Block-based programming enables lecturers to introduce learners in a step-by-step process to a text-based programming syntax: First they only read the code, then they are asked to comment it and finally they start to write (small) parts of the program in the text-based language. This approach enables teachers to associate one problem with multiple levels of task complexity to teach a heterogeneous group of learners.

3 Block-based programming interface for the ESP32

Due to the major benefits that have been evaluated in a detailed comparison between four Block-based Programming IDE's (*Edublocks for ESP32*, *TUNIOT for ESP32*, *Blocklyduino* and *Ardublockly*) [6, p. 12–14], *Ardublockly* has been selected as framework to integrate the ESP32 microcontroller and implement new blocks based on the new possibilities with this more recent microcontroller. After this evaluation on 18 characteristics within six major categories, the major benefit of *Ardublockly* was revealed to be its outstanding didactic qualification, as its view always shows blocks and text-based code simultaneously and it highlights the generated text-based code for each added block in real-time [6, p. 12–14]. As *Ardublockly* was released under the Apache 2.0 license it is open for any modification. Besides the integration of the ESP32 Board into *Ardublockly* **the implementation consists of eleven new modules which were developed with 27 new blocks**. Each implementation was done based on specified requirements. Additionally the advantages for school education were revealed for each implementation. After the integration of the ESP32 with all its interfaces into *Ardublockly*, the primary interest was to develop blocks to use the new features provided by the hardware. *Ardublockly* neither offered a possibility to establish a network communication nor provided implementations of network protocols. Therefore efforts were spent on **one block to connect the microcontroller to an existing WiFi network, and another to switch the microcontroller into access-point mode** and create its own WiFi network.

On top of this basic WiFi connector implementation any network protocol can be implemented. As IoT is a central evolution of the last decade within the field of computer science, the objective was to enable all teachers and lecturers to include real-life IoT scenarios into their classes. Therefore the IoT Network Protocol **MQTT was fully implemented by establishing four new blocks**. This includes the publish feature from the MQTT protocol (e.g. to publish sensor values) as well as the subscribe feature of MQTT to constantly receive new information on a specific topic.

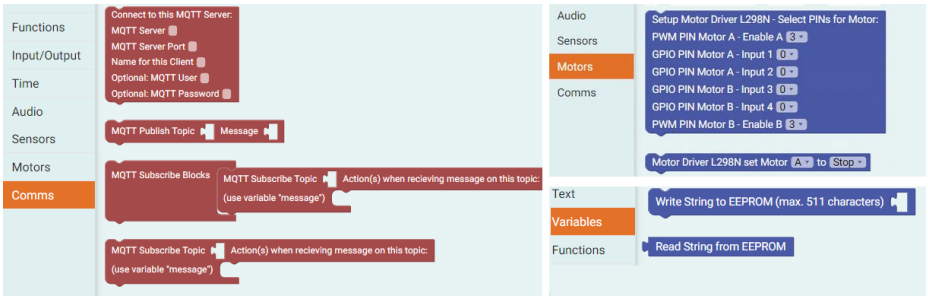

Fig. 1. Part of the introduced blocks. Details to all block implementation can be found in [6, p. 19–31]

To utilize the Block-based IDE with the ESP32 for applications in the field of robotics, features to handle robot movements are key. Admittedly in Ardublockly blocks to handle standard DC motors were missing. Therefore **two new blocks were developed to enable the Motor Driver L298N** to run DC Motors by supporting a two-channel operation. The standard types of movement – forward, backward, stop – have been implemented directly to avoid low-level pin handling. To allow full control of movement a speed regulation for the motors is necessary. This utilizes the technique of pulse width modulation (PWM) to handle the power-flow. Therefore another **3 blocks were developed to implement the generic PWM functionality**.

A robot uses sensors to receive information from its surroundings and to enable orientation. One communication channel to these sensors is the I^2C bus. To facilitate error handling with cabling for sensors that use the I^2C bus, an **I^2C Scanner was implemented using only one block**. It detects the connected I^2C devices and prints their unique device addresses to the serial bus.

Typically the output of microcontroller programs developed in Ardublockly is printed to the serial bus, which is observed by the serial monitor of the Arduino IDE. This turns out to be problematic for robotic scenarios as soon as the robot starts moving. Therefore the **OLED Display Adafruit SSD1306 was included into the IDE by developing three new blocks** as a new option to deliver outputs to the user.

Another characteristic of microcontrollers is that they store all of their variables and actual GPIO Pin statuses within their RAM. This leads to an essential problem when it comes to a (short) power outage at the microcontroller because all data and status will be gone and the program will restart from scratch. The decision to equip learners with the possibility to store information permanently (e.g. GPIO Pin state) and let them recover this information after a reboot led to the **development of two new blocks for EEPROM management**. The blocks allows a string of up to 511 characters to be stored into the EEPROM as the ESP32 emulates the EEPROM with 512 Byte of its flash memory.

All developments together form the new block-based visual programming IDE extension to Ardublocky. The Windows and Linux release together with its documentation can be found at https://github.com/pati5000/Ardublockly-ESP/. In addition to this documentation, two specific manuals for teachers were created, a compact, single-page manual for teachers (*Ardublockly ESP Quick Start Guide*) in German and English and an advanced manual in English that enables teachers to easily develop new blocks.

4 Bringing Swarm Robotics into schools using the ESP32

The IDE is developed to be suitable for beginners as well as advanced learners. The following exercise in the field of swarm robotics should demonstrate how powerful the IDE is simply by combining the implemented modules. Swarm robotics was chosen as it is based on interactive communication between robots

and provides beneficial new directions for robotics in education. Moreover, swarm robotics is very rarely discussed in the field of computer science education.

The ultimate goal for learners in this example is to build a self-balancing seesaw such that a variable number of robots are distributing their weights to keep the seesaw in balance.

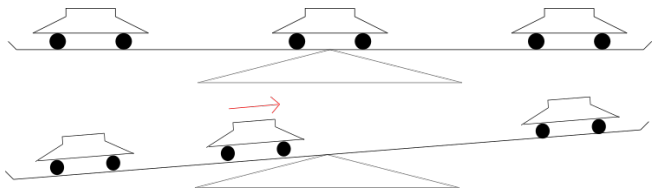

Fig. 2. Seesaw example setup similar to the setup in this video [14]

The seesaw itself is equipped with a ESP32-DevKitC SoC that operates a 3-axis gyroscope and acceleration sensor (model MPU-6050). This microcontroller controls the positioning of the seesaw. It reads the positioning information from the sensor and sends it via a WiFi Network to an MQTT broker using the MQTT protocol. Another PC in the network is supposed to act as an MQTT broker or it is assumed that a MQTT cloud service is used.

The robots subscribe via MQTT to receive this information. Each of them is equipped with a ESP32-DevKitC SoC, a minimum of six infrared sensors, a 3-axis gyroscope and acceleration sensor (model MPU-6050), a L298N motor driver, as well as DC motors and wheels where the quantity depends on the individual design. The infrared sensors are needed to enable the robots to detect the edges of the seesaw. The gyroscope and acceleration sensor is only needed to determine on which side of the seesaw the robot is located. The motor driver and the motors are used to move the robot in specific directions to distribute its own weight. Based on the information of the gyroscope and acceleration sensors the swarm intelligence will calculate and control the move of the robots.

The swarm intelligence is implemented in one program which will be provided to all robots. The robots will communicate on three MQTT topics:

- Subscribe the status of the gyroscope and acceleration sensor in the seesaw (acts as a single source of truth),
- Publish the seesaw and robot status after each cycle of move,
- Negotiate who will take the next action.

The challenge for learners is to develop a protocol for this swarm communication and decision making of the swarm. Only one robot should move at a given time. After a move there should be another cycle of measurement and negotiation before there is a new move.

Most probably it will be required to introduce the learners to the field of swarm intelligence. Therefore lecturers can run the workshop developed by Stovold [12, p. 203–204]. In this workshop the learners try to find a real-life swarm

intelligence algorithm based on the example of the behaviour of wild fireflies. The goal is to synchronize their flashing with each other to attract females. The learners should be allowed some time to discuss their ideas in small groups and ideally demonstrate their solutions to the whole class. This will prepare them for the discussion of the algorithm they need to develop for this robot balancing task.

5 Conclusion

The ESP32 and Ardublocky with its ESP extension is an attempt to boost education in computer science, physical computing and especially educational robotics. A modern, powerful and affordable microcontroller and an IDE which is optimized for educational purposes has been developed as a state-of-the-art didactic environment for learners at multiple (class) levels to learn programming at individual depth and speeds by experimenting on physical computing projects.

Learners benefit from block-based programming as algorithmic thinking is developed without the tedious act of learning the syntax from a specific, text-based programming language.

By providing a workshop and an advanced programming example on swarm robotics, this paper picked a field that is rarely discussed in computer science education but is highly visible in nature. This paper also contributes to the development of more holistic, motivating and innovative computer science lectures. Well-designed computer science education is an effective way to support learners to gain relevant competences for their future.

References

1. Austin, J., Baker, H., Ball, T., Devine, J., Finney, J., de Halleux, P., Hodges, S., Moskal, M., Stockdale, G.: The BBC micro:bit – from the UK to the World. Communications of the ACM (4 2019)
2. Chandler, P., Sweller, J.: Cognitive load theory and the format of instruction. Faculty of Education - Papers **8** (12 1991). https://doi.org/10.1207/s1532690xci0804_2
3. Fuchs, K.J., Milicic, G.: Algorithmisches Denken im anwendungsorientierten Unterricht. Didaktik der Naturwissenschaften. Neue Horizonte in Biologie, Geometrie und Informatik (2016)
4. Hughes, J.M.: Arduino: A Technical Reference: A Handbook for Technicians, Engineers, and Makers. O'Reilly Media, Inc. (2016), media.digikey.com/pdf/Data%20Sheets/O'Reilly_PDFs/Arduino_A_Technical_Reference_9781491921760.pdf
5. Kortuem, G., Bandara, A.K., Smith, N., Richards, M., Petre, M.: Educating the Internet-of-Things generation. Computer **46**(2), 53–61 (2012)
6. Lamprecht, P.: Entwicklung eines Block Programming Interface für den Mikrocontroller ESP32 zum Einsatz im modernen Informatikunterricht (2019), http://dx.doi.org/10.13140/RG.2.2.13935.46249
7. Maier, A., Sharp, A., Vagapov, Y.: Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things. In: 2017 Internet Technologies and Applications (ITA). pp. 143–148. IEEE (2017)

8. Micro:bit Educational Foundation: Micro:bit Hardware (2020), https://tech.microbit.org/hardware/
9. Perenc, I., Jaworski, T., Duch, P.: Teaching programming using dedicated Arduino Educational Board. Computer Applications in Engineering Education (2019)
10. Przybylla, M., Henning, F., Schreiber, C., Romeike, R.: Teachers' Expectations and Experience in Physical Computing. In: International Conference on Informatics in Schools: Situation, Evolution, and Perspectives. pp. 49–61. Springer (2017)
11. Przybylla, M., Romeike, R.: Empowering learners with tools in CS education: Physical computing in secondary schools. it-Information Technology **60**(2), 91–101 (2018)
12. Stovold, J., Powell, S.: Teaching Object-Oriented Programming in Secondary Schools Using Swarm Robotics. In: Moro, M., Alimisis, D., Iocchi, L. (eds.) Educational Robotics in the Context of the Maker Movement. pp. 201–204. Springer (2020)
13. Wintrop, D., Wilensky, U.: To block or not to block, that is the question: students' perceptions of blocks-based programming. In: Proceedings of the 14th International Conference on Interaction Design and Children. pp. 199–208. Marina Umaschi Bers (2015)
14. Íñiguez, A.: Swarm Technology, Ants, Robots, IoT and Parallel Processing (2016), https://www.youtube.com/watch?v=ElEHN8tHJnc