*ICIC Express Letters*
*Part B: Applications*
Volume **11**, Number **9**, September **2020**

ICIC International ©2020 ISSN 2185-2766
pp. 815–822

# DEVELOPMENT OF WEB-BASED ARDUINO SIMULATOR SUPPORTING DEBUG FUNCTION AND STATUS VISUALIZATION

Kozo Horiuchi[1], Takanori Matsuzaki[2] and Hiroshi Shiratsuchi[2]

[1]Gururi Co., Ltd.
MATSUNAGA bldg. 5F MIKAGE1881, 2-1-7 Uomachi, Kokurakita, Kitakyusyu
Fukuoka 802-0006, Japan
kozo.horiuchi@gururi.co.jp

[2]Graduate School of Humanity-Oriented Science and Engineering
Kindai University
11-6 Kayanomori, Iizuka, Fukuoka 820-8555, Japan
{ takanori; sira }@fuk.kindai.ac.jp

Received February 2020; accepted May 2020

Abstract. *Embedded control systems are applied for various fields such as manufacturing, industry, and recently as an IoT for agriculture and medicine. In our department subject of "system programming", students learn to the Arduino as follows: (i) Turn on the LED, (ii) Convert musical pitch to the frequency and sound a buzzer, (iii) Change internal state by a switch element, and (iv) Supervise and manage complex state transitions. This paper proposes an Arduino computer simulation system to understand the above topics. The proposed system has functions such as "Program debug", "Arduino simulator", "Circuit validation", and "Display for the status of each IO port and internal state transition". By implementing these functions, the proposed system can be visually utilized for "Sensor input based on operating specifications" and "Device control using timers and switch interrupts" which are difficult to learn only with the Arduino. At the same time, automatic program coding based on a state transition diagram due to visually understanding program execution and transition internal status is also developing.*

**Keywords:** Arduino, Debugging, State transition diagram, Engineering education, Web application, JavaScript

## 1. Introduction.

Embedded control systems are applied for various fields such as manufacturing, industry and recently as an IoT (Internet of Things) for agriculture and medicine [1, 2, 3, 4]. However, to understand them, knowledge in a wide range of fields such as electrical theory, electronic circuits, control theory, and information communication is required. In our department, students learn about electric and electronic circuit theory and circuit design in the lecture "Electric circuit" and "Electronic circuit", and learn about actual circuit making and measurement using the various measuring instrument in "Electric engineering experiment" as practice. At the same time, they are learning the basics of C programming language such as variable type, control structure, function and pointer in the subjects of "Programming" and "Data structure and algorithms" [5, 6]. System programming is an integrated course subject of electronic circuits and programs. Concretely, in order to learn about the following 4 topics, the circuit designed on the breadboard with electronic parts, resistors, and LEDs (Light Emitting Diode) is connected to Arduino UNO and is controlled by the program description. (i) Turn on the LED, (ii) Convert musical pitch to the frequency and sound a buzzer, (iii) Change internal state by a switch element, and (iv) Supervise and manage complex state transitions. The exercises in this course use the Arduino official development environment "Arduino IDE (Integrated

DOI: 10.24507/icicelb.11.09.815

Development Environment)"; however, there is not debugging equivalent [7, 8]. Similar to software development, debugging functions such as breakpoints, step execution and reference of internal variables are also useful in hardware programs [9]. Especially, in this lecture, since circuit design and program development are required to perform in parallel, it is difficult to distinguish whether the cause of the troubles encountered is hardware or software. For example, it is difficult to distinguish circuit implementation errors such as "Polarity of the LED" and "Pin wiring" from software declaration and program reference errors such as "Variable" and "Pin Number" [10, 11].

This paper proposes an Arduino computer simulation system to understand the above topics. The proposed system has functions such as "Program debug", "Arduino simulator", "Circuit validation", and "Display for the status of each IO port and internal state transition". By implementing these functions, the proposed system can be visually utilized for "Sensor input based on operating specifications" and "Device control using timers and switch interrupts" which are difficult to learn only with the Arduino. At the same time, automatic program coding based on a state transition diagram due to visually understanding program execution and transition internal status is also developing.

Specifically, Section 2 describes the composition of the learning material in use and, Section 3 progresses with the learning and current issues. Section 4 provides an outline of the proposed simulator and the features that will be implemented. Section 5 summarizes the paper.

## 2. **Composition of Learning Materials.**

The "system programming" course offered in this department uses the Arduino UNO as teaching material and conducts exercises. Figure 1 shows the main parts of the learning system that students are using. "Arduino UNO" at the red frame and "400 holes breadboard" at the yellow frame are fixed and used on "Arduino and Breadboard Holder" at the green frame in the figure.

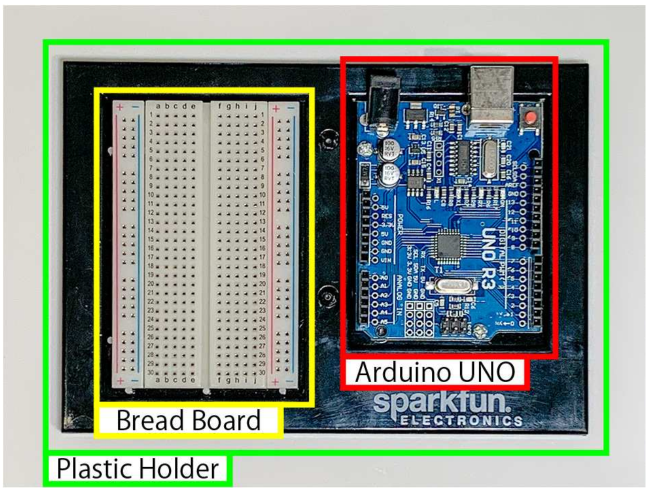


FIGURE 1. Overview of learning materials for Arduino

Power is supplied to the Arduino UNO from a USB (Universal Serial Bus) cable that is connected during development. The system operates at a voltage relative to 5 [V].

Connect each port of the Arduino UNO to the breadboard using jump wires. Figure 2 shows the electronic components used throughout the exercise. The students use 20 jump wires of different lengths shown in the red frame. The electronic components in the green frame are "resistor", "LED", "piezo buzzer", "phototransistor" and "tact switch". The students combine these electronic components with the Arduino UNO programs to perform the exercises.

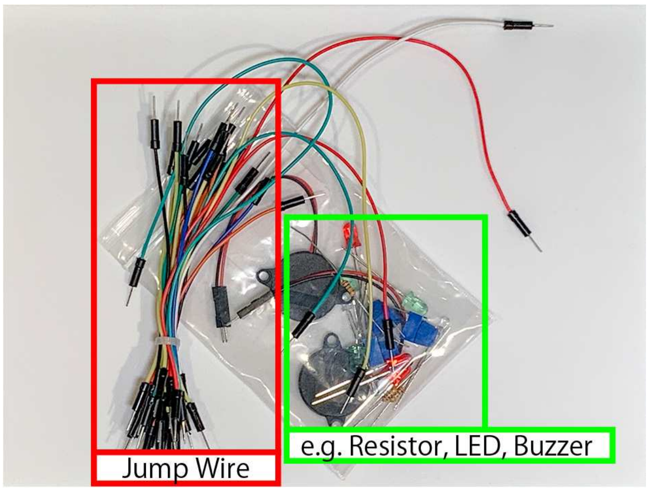


Figure 2. Jump wires and electronic components

3. **How to Progress with the Learning.** In the "system programming" subject, 1 to 2 questions are asked per lecture. Students create and submit programs and circuits that meet the specifications in time. In the first half of the lecture, the lecturer explains the content necessary to solve the question. Since the subject is aimed at students with knowledge of C language, the lecturer will focus on the differences between writing in C language and Arduino-specific descriptions, functions, and function usage. The remaining time is allocated to the exercise time.

3.1. **Contents of the present subject.** The subject is divided into four main sections. In "(i) Turn on the LED", the program and circuit are introduced to understand the output ports of the Arduino and to make the LEDs illuminate.

Start with simple lighting, and then create a program that gradually increases the difficulty by "Flashing at various intervals" and "Lights alternately using multiple LEDs".

Next, in "(ii) Convert musical pitch to the frequency and sound a buzzer" they apply knowledge in (i). They control programmatically the ON and OFF times of the voltage applied to the piezo buzzer connected to the port, and confirm that an arbitrary pitch is emitted. Students also learn how to stop a program that proceeds sequentially to control the ON and OFF times. Normally, the Arduino UNO runs very fast compared to the human sense, so we need to use proper stopping. After learning these points, students create a program to play typical Japanese children's songs.

In "(iii) Change internal state by a switch element", students learn how to use input ports and interrupt handling. The Arduino UNO can handle interrupts only on certain ports (D2, D3) and can specify when to receive interrupts as LOW (port voltage LOW), CHANGE (port voltage change), RISING (port voltage LOW to HIGH), and FALLING (port voltage HIGH to LOW). This lecture uses the pull-up circuit shown in Figure 3. Specify the interrupt timing FALLING because the circuit is short-circuited by the switch ON. When the switch is pressed, interrupt processing is performed. For example, "increase the number of counters" and "change a variable that represents a state" are processed during interrupt processing. When all the interrupt processing is completed, the processing returns to the original processing. Students learn "How to Change the Internal State of a Program" through switching and interrupt processing and then create "How the switch toggles 2 LEDs" and "Changing the LED lighting cycle according to the internal state" programs. In addition, students will learn about chattering phenomena caused by physical switches, as shown in Figure 4, and also learn software solutions.

Finally, in "(iv) Supervise and manage complex state transitions", students learn using an analog port and phototransistors. Figure 5 shows a circuit that converts the intensity

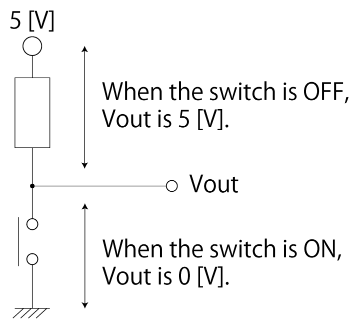


FIGURE 3. Diagram of pull-up circuit

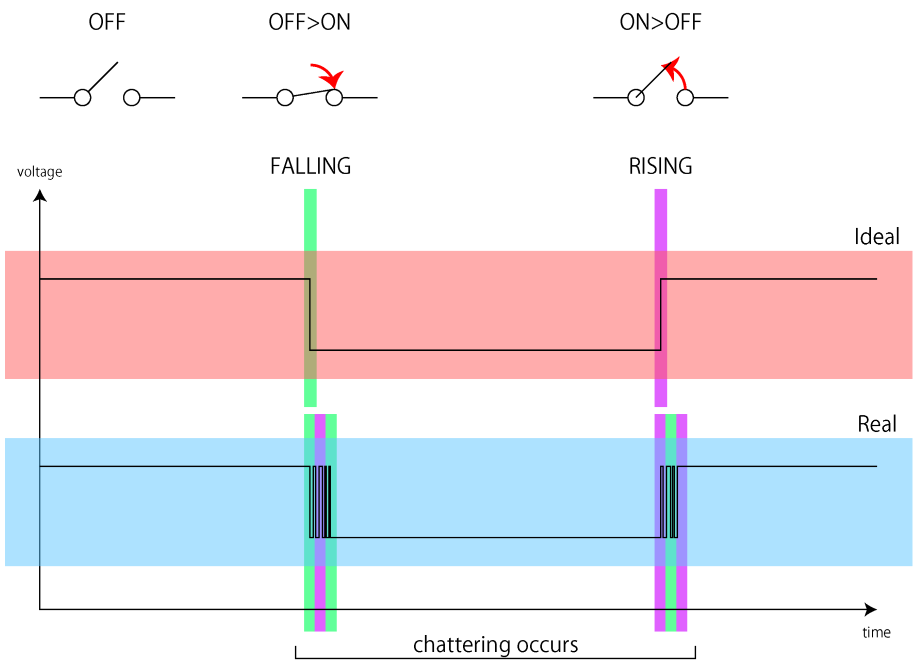


FIGURE 4. Example of chattering phenomena caused by physical switches

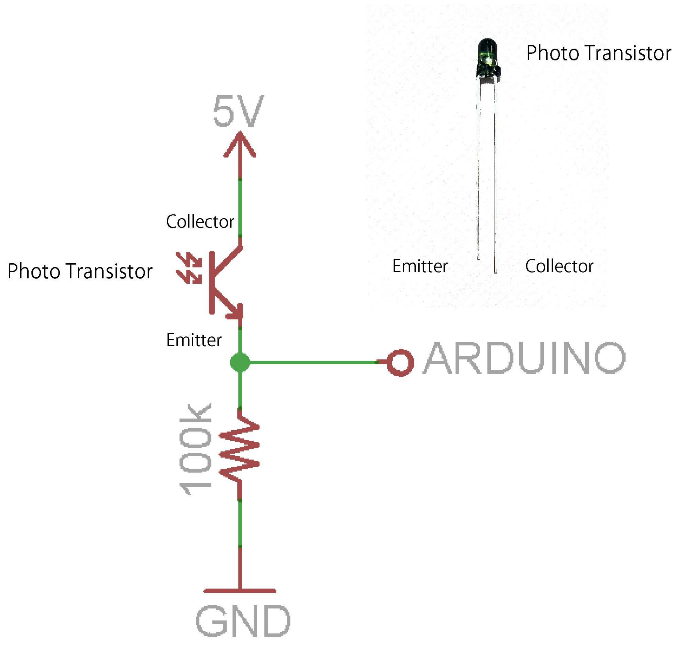


FIGURE 5. A circuit sample converting the intensity of visible light into a voltage

of visible light into a voltage. Connect this circuit to the analog port and create a program based on the A-D (Analog-Digital) converting the digital value. Students learn how to change the internal state of a program by events that are not physical operations (such as the switch).

3.2. **Current issues.** Learning items (i) and (ii) tend to be easier to understand because the results are directly expressed in light or sound. However, in item (iii), the student cannot directly confirm "Switch Interrupt Relationship" and "Internal state after interruption". It is necessary for students to make an analogy from the lighting of LEDs, which makes it difficult to understand. In item (iv), it is also difficult to clearly understand the state of progress by combining internal states. In order to solve these problems, it is necessary to have a function to grasp the status of each input/output port while visually grasping the operation status of the program.

## 4. **Outline of the Proposed Simulator.**

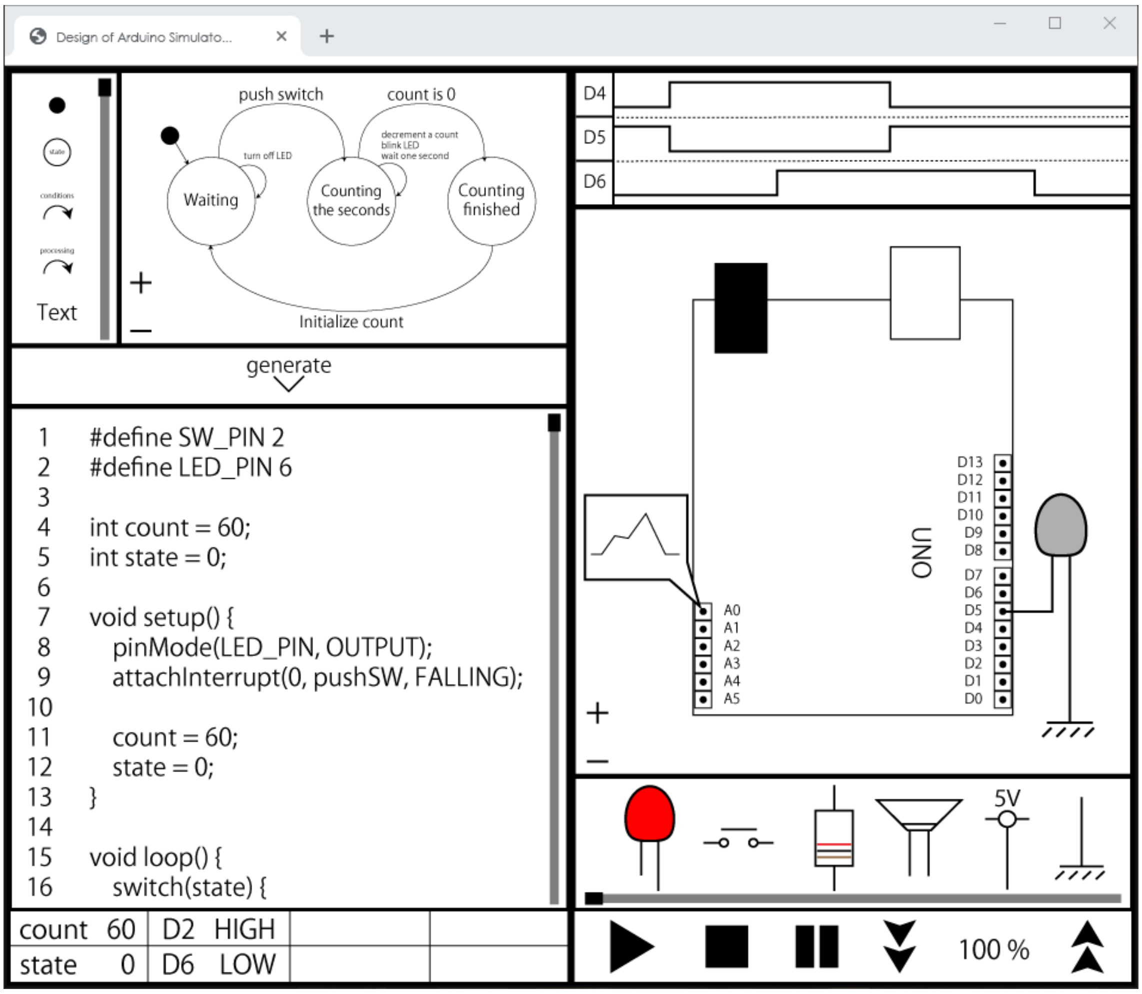
The simulator assumes a Web application is running in a browser. In recent years, the network speed and JavaScript execution speed are fast. We considered implementing it in a native application, but since it does not necessarily depend on the OS (Operating System), we adopted a Web application. Next, in terms of functionality, the following items are important for solving current problems.

- A visual sense of the internal state
  - Using state transition diagram
  - Understand internal parameters (Variable)
- Visualization of the port voltage
  - Check output port status
  - Input port state given by any state change
    - Switch, Sensor

As a general debugging function, it is also necessary to pause and resume a program at an arbitrary time. In the simulation, it is difficult to operate at the same speed as the actual machine. However, it runs fast enough compared to the human senses. Therefore, in order to check various items effectively, a function to adjust the execution speed is required. The following are the specifications currently assumed.

- Digital output port
  - D4 to D7
- Digital input port (interrupt)
  - D2, D3
- PWM (Pulse Width Modulation) (Analog output)
  - D5, D6
- Analog input port
  - A0 to A5
- Standard function for Arduino
  - setup, loop
  - pinMode
  - digitalRead, digitalWrite
  - analogRead, analogWrite
  - attachInterrupt
  - delay, delayMicroseconds
- Run sampling time
  - 10 milliseconds
- Run rate of delay
  - 100 to 10000 %

Figure 6 shows the current simulator screen. Describe each function in the following sections.



```
#define SW_PIN 2
#define LED_PIN 6

int count = 60;
int state = 0;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  attachInterrupt(0, pushSW, FALLING);

  count = 60;
  state = 0;
}

void loop() {
  switch(state) {
```

| | | | |
|---|---|---|---|
| count | 60 | D2 | HIGH |
| state | 0 | D6 | LOW |

Figure 6. State transition diagram of "60-second timer"

4.1. **Sketch simulation method.** The sketch simulator is implemented as a server-side compiler (language C) or client-side interpreter (JavaScript) and simulates a sketch which is student programmed. The proposed system runs an existing compiler on a Web server to perform syntax checking, compilation, and execution. The browser sends the sketch written by the user and the signal input to the port to the server and reflects the result of execution.

4.2. **Checking internal parameters.** Variables used in sketches are listed (Figure 6 Bottom Left). In the example of "60-second timer", because the count variable is decremented by 1 every 1 second, it can be checked at the normal execution speed. However, some implementations may consider counting every 1 millisecond, but variables may change too quickly and be difficult to verify. In such cases, the execution speed can be adjusted (Figure 6 Bottom Right).

4.3. **Port voltage visualization.** When connecting microcomputers and circuits through ports, it is very important to understand the state of each port. The proposed system can check the high (HIGH:1) or low (LOW:0) status of any selected port as a timing chart (Figure 6 Top Right).

4.4. **Providing a signal to port.** In order to verify that a program operates correctly with the behavior of switch or the input of sensor element, the proposed system can change input value of ports at any timing (Figure 6 Middle Right). In the example of "60-second timer", when the start switch is pressed, the state changes from the initial state to the countdown state, and the timer begins counting 60 seconds. To verify the program satisfies the specifications, it is necessary to provide several patterns sequences to these ports. The system also has to support continuous voltage change due to simulating a sensor input.

4.5. **Creating a sketch with a state transition diagram.** In many cases, a programmer who is familiar with programming can write a program while checking specifications. However, it is difficult for students who have just started learning. Therefore, in the "system programming" subject, we taught students to draw a state transition diagram from the specifications of the exercise. Drawing a state transition diagram clarifies the states present in the specification and the processing to be performed between them, the timing of the state change, and the conditions. Figure 7 shows a state transition diagram based on the example of "60-second timer". The proposed system can generate a sketch template from a drawn state transition diagram. Because the generated sketches and state transition diagrams are linked, parts corresponding to current running program blocks and the state transition diagram are emphasized when simulated on the system. Thus, states in the program and state transition diagram states can be visually displayed. This visual display makes it possible to check in detail whether there is any difference between the actual specifications and the behavior of the created program.

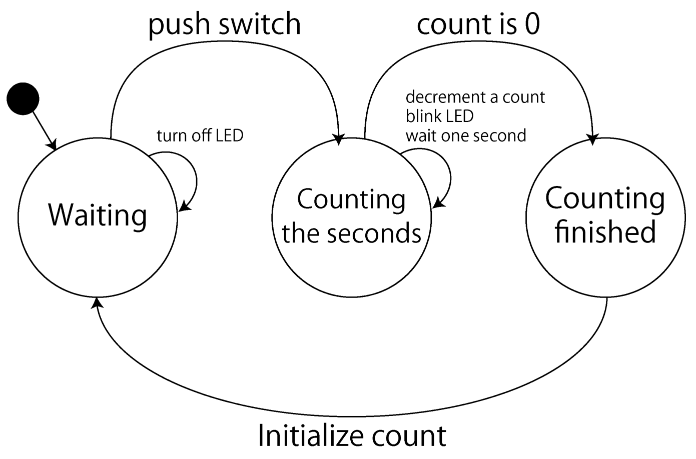


Figure 7. State transition diagram of "60-second timer"

4.6. **Improvement through the proposed simulator.** By implementing the functions described in the previous section, students will be able to understand the internal state, which was difficult to learn with Arduino IDE alone. Specifically, visual understanding is possible from the internal state displayed by the breakpoints and step execution provided as a debugging function in the proposed simulator. The program code in the automatically generated sketch template corresponds to each state in the state transition diagram. For this reason, if each state and the transition condition triggered by switches or sensors are correctly expressed as a state transition diagram, it is possible to generate a program code that meets the system specifications for exercises. Therefore, it is considered that the program development becomes possible while checking the internal state and transition conditions in cooperation with the simulator.

5. **Conclusions.** In this paper, we proposed an embedded control learning system for education as a Web application and described the outline, procedures, and problems of the learning system using Arduino. Then, we considered the specifications of the simulator available in the browser as a mechanism to compensate for the problems and considered the necessary functions.

Since the proposed system could simulate relations of the Arduino and the circuit, the students can observe the state of each port and the internal state change when switches status and sensor inputs have changed. Furthermore, this system also has debugging functions that are not included in the "Arduino IDE" standard development environment. These functions make it possible to pause, breakpoint, step by step execution, resume, and adjust the execution speed at any time. Therefore, in the proposed system, internal variable and state transitions that are difficult to observe in actual systems can be displayed visually. Not only, moreover, sketch templates can be generated from state transition diagrams, but also state transitions and processing flows can be visualized.

In future research, we plan to implement a simulator and have the students use it to investigate improvements in proficiency and learning efficiency.

## REFERENCES

[1] P. Gupta, D. Agrawal, J. Chhabra and P. K. Dhir, IoT based smart healthcare kit, *Proc. of Int. Conf. Comput. Tech. Inf. Commun. Technol. (ICCTICT 2016)*, pp.237-242, 2016.

[2] A. M. Ghosh, D. Halder and S. K. A. Hossain, Remote health monitoring system through IoT, *Proc. of the 5th Int. Conf. Inform. Electron. Vis.*, pp.921-926, 2016.

[3] P. K. Naik, A. Kumbi, V. Hiregoudar, N. K. Chaitra, H. K. Pavitra, B. S. Sushma, J. H. Sushmita and P. Kuntanahal, Arduino based automatic irrigation system using IoT, *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol.2, no.3, 2017.

[4] J. C. Negrete, E. R. Kriuskova, G. De J. L. Canteñs, C. I. Z. Avila and G. L. Hernandez, Arduino board in the automation of agriculture in Mexico, a review, *International Journal of Horticulture*, vol.8, no.6, pp.52-68, 2018.

[5] A. Bashir, M. Alhammadi, M. Awawdeh and T. Faisal, Effectiveness of using Arduino platform for the hybrid engineering education learning model, *IEEE ASET 2019*, pp.1-6, 2019.

[6] D. Belfadel, M. A. Rodriguez, M. Zabinski, R. Munden and J. Cavallo, Use of the Arduino platform in fundamentals of engineering, *ASEE 2019*, 2019.

[7] N. Mitsunaga, An interpreted language with debugging interface for a micro controller, *IEEE GCCE 2012*, pp.115-119, 2012.

[8] Y. Torroja, A. Lpez, J. Portilla and T. Riesgo, A serial port based debugging tool to improve learning with Arduino, *2015 Conference on Design of Circuits and Integrated Systems (DCIS)*, pp.1-4, 2015.

[9] T. Nitta, T. Furukawa and M. Ohchi, Proposal of environment for learning using the virtual single board computer "x80", *IEEJ Trans. Electronics, Information and Systems*, vol.125, no.1, pp.128-133, 2005.

[10] M. A. Rubio, C. M. Hierro and Á. P. D. M. Pablo, Using Arduino to enhance computer programming courses in science and engineering, *Proc. of EDULEARN13*, pp.5127-5133, 2013.

[11] R. Chancharoen, A. Sripakagorn and K. Maneeratana, An Arduino kit for learning mechatronics and its scalability in semester projects, *2014 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pp.505-510, 2014.