



## Physical Computing Modul for Informatics Learning

Muhamad Afif Effindi<sup>1</sup>, Wuritri Handayani<sup>2</sup>

<sup>1,2</sup> Universitas Trunojoyo Madura, Bangkalan, Indonesia

Correspondence: [mafif.effindi@trunojoyo.ac.id](mailto:mafif.effindi@trunojoyo.ac.id)

### ABSTRACT

Informatics is a relatively new subject in Elementary and Secondary Education in Indonesia. Referring to the Guidelines for the Implementation of Informatics Content/Subjects of the 2013 Curriculum and related regulations and circulars, the subjects implemented since the 2019-2020 school year must have a School Repository. The Repository contains a collection of Learning Plans, Teaching Materials, and Portfolios that record the implementation of Informatics subjects. The challenge in developing subject material is the need to develop a combination of hardware and software for informatics learning class. This research aims to develop a Physical Computing Teaching Modul in order to answer the challenge. This research method includes a Literature Review, Material Design in the Teaching Module, and the creation of a Teaching Module with attractive packaging suitable for use in class. The research output is a Physical Computing teaching Modul.

© 2024 Universitas Pendidikan Indonesia

### ARTICLE INFO

#### Article History:

Submitted/Received 13 Jan 2024

First Revised 05 Feb 2024

Accepted 02 May 2024

First Available online 06 Jun 2024

Publication Date 06 Jun 2024

#### Keyword:

Physical Computing,

Arduino,

Informatics Learning

## 1. INTRODUCTION

Indonesia has vast potential in the field of information and communication technology. This can be confirmed in terms of the number of internet users reaching 143 million, the number of digital start-up companies reaching 992 companies, and the number of Unicorns reaching five companies (Effindi, 2020). From the perspective of this potential, implementing Informatics subjects that have been implemented since the 2019-2020 school year is the right step for many parties. This is a form of responsiveness or quick response when facing the Industrial Revolution 4.0.

Indonesia's Ministry of Education and Culture has also issued Guidelines for Implementing Informatics as Content or Subjects in the 2013 Curriculum. The guidelines state the need for schools or Subject Teachers' Conferences (MGMP) in a region to have a repository to store resources for implementing Informatics subjects. One of the resources stored in Teaching Materials discusses the study of Informatics subjects.

The components of Knowledge or Informatics Studies for elementary and secondary education include Information and Communication Technology (ICT), Computational Thinking (BK), Computer Engineering (TK), Computer/Internet Networks (JKI), Data Analysis (AD), Algorithms and Programming (AP), Social Impact of Informatics (DSI), and Cross-Field Practices (PLB) (Amalia et al., 2023; Effindi, 2020, 2022; Kartiansyah et al., 2021). The components of Informatics knowledge are presented in Figure 1.1. Some of the Informatics knowledge components materials contain abstract concepts, including programming.

The basic concepts of computer programming are a challenge to teach to students. As reported by Chung and Lou (2021) and (Vasilchenko et al., 2020), that basic programming concepts are abstract concepts complex for students new to coding to understand. Meanwhile, (Roumen & Fernaus, 2021) explains another challenge in teaching programming: the difficulty of finding the right tools to use in assignments or group work on programming materials. It is stated that using laptops for programming assignments is more appropriate for each student in individual assignment settings but is challenging to use for collaborative work (Roumen & Fernaus, 2021).

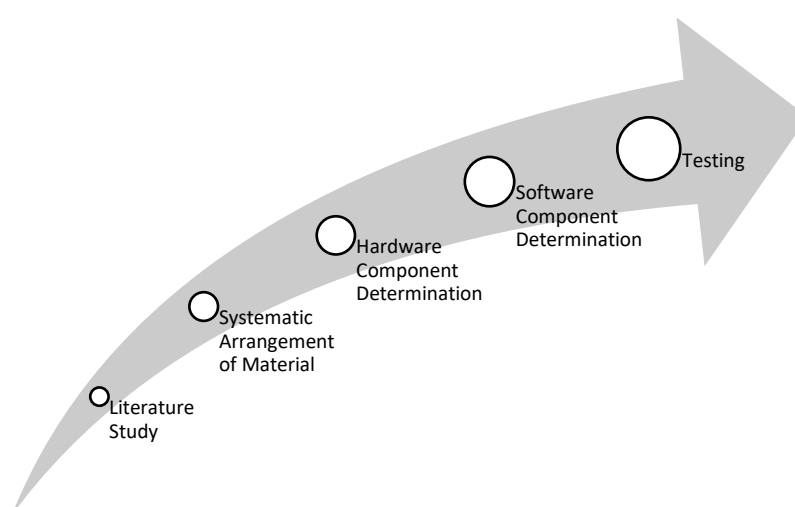
The challenges in teaching programming concepts have been studied to provide solutions. One of them is providing a variety of material packaging and teaching strategies. Among the material packaging that can be selected is the application of Physical Computing to programming concept materials. Physical Computing combines hardware and software, involving physical input, such as movement, temperature, and light (Hodges et al., 2020). The term Physical Computing was introduced by O'Sullivan and Igoe in 2004 (Roumen & Fernaus, 2021). The term of physical computing also mentioned as Tangible Computing by Horn and Bers (2019). Physical Computing refers to using hardware and software in teaching to provide students with an understanding of computing systems that can be physically observed.

The selection of Physical Computing as a tool for learning in this research is due to various reports and studies stating that the application of Physical Computing can increase student motivation to learn to understand programming concepts (Chung & Lou, 2021), enhance cooperation and collaboration between students, as well as increase creativity (Katterfeldt et al., 2018; Roumen & Fernaus, 2021), improve digital skills in terms of programming abilities (Vasilchenko et al., 2020), and improve motivation on result-oriented in computational thinking (Alden & Tramonti, 2020; Jormanainen & Tukiainen, 2020).

Considering the positive impact of Physical Computing on programming learning, which is an integral part of Informatics Learning, this research proposes the development of a Physical Computing-based Teaching Module. The development of this Teaching Module is a significant contribution, as explained in the Informatics Content/Subject Implementation Guidelines, that schools implementing Informatics content/subjects need to compile a repository containing teaching materials ready to be used in classroom activities.

## 2. METHODS

This research consists of several stages, including starting with a literature review, compiling a systematic material, determining hardware components, determining software components, and testing. Figure 1 shows the stages of research carried out in this study. The literature review stage is carried out to obtain information related to several things, including the method of compiling teaching modules, studying hardware components that can be used in learning, and studying software components that can be used in learning. The next stage is the development stage of a Physical Computing-based Teaching Module. Developing a Teaching Module consists of several sub-stages, including compiling a systematic material, determining hardware components, determining software components, and designing a teaching module. The sub-stage of compiling materials is intended to review and determine the study and discussion in the Teaching Module.



**Figure 1.** Research stages

The hardware component determination sub-stage is intended to examine and select hardware that can be used in the Physical Computing-based Teaching Module. The software component determination sub-stage is used to determine, compile, and present the software used in the Physical Computing-based Teaching Module. The next sub-stage is the design of the teaching module. The work aims to designing this teaching module specifically to make the packaging appear attractive. All stages of this research will be continued with the testing stage.

## 2.1 The Design of Physical Computing Modul

Physical Computing combines hardware and software to form a computing system that receives physical input. Considering this concept, the Teaching Module developed in this study needs to undergo studies and designs for several aspects, including Material Systematics, Hardware Components, Software Components, and Teaching Module Design.

## 2.2 Systematics of Material

Table 1 shows the systematics of the material included in the Physical Computing-based Teaching Module developed in this research.

**Table 1.** Systematics of Material

No	Modul Unit	Content	DBL Arduino Programming Course Content
1	1-2	Unit 1: A preliminary study on program design and pet wearable device	<ol style="list-style-type: none"> <li>1) Pretest questionnaire</li> <li>2) Introduction of programming</li> <li>3) Concept establishment of project development</li> <li>4) Literature collection and reading</li> <li>5) Grouping plan</li> <li>6) Pet wearable devices</li> </ol>
2	3-5	Application of microcontroller program	<ol style="list-style-type: none"> <li>1) C++ language architecture</li> <li>2) Arduino architecture</li> <li>3) Resource utilization of Arduino online platform</li> <li>4) Program conception and reorganization</li> <li>5) Programming teaching</li> </ol>
3	6-8	LED and sensing switches	<ol style="list-style-type: none"> <li>1) LED application</li> <li>2) Light sensing, tilt sensing</li> <li>3) Programming practice</li> <li>4) Design thinking teaching</li> </ol>
4	9-11	LED and temperature and humidity sensing	<ol style="list-style-type: none"> <li>1) Temperature and humidity module</li> <li>2) WIFI transmission</li> <li>3) Programming practice (Description—Execution—Reflection—Debugging—Description)</li> <li>4) Design thinking practice</li> <li>5) (Discover—Define—Develop—Deliver)</li> </ol>
5	12-16	Pet wearable device project	<ol style="list-style-type: none"> <li>1) Accessory design of pet wearable devices</li> <li>2) Circuit practice</li> <li>3) Programming utility (Description—Execution—Reflection—Debugging—Description)</li> <li>4) Design thinking application</li> <li>5) (Discover—Define—Develop—Deliver)</li> </ol>
6	17-18	Project presentation	<ol style="list-style-type: none"> <li>1) Project practice report writing</li> <li>2) Design concept exchange</li> <li>3) Publication and display of achievements</li> <li>4) Posttest questionnaire</li> </ol>

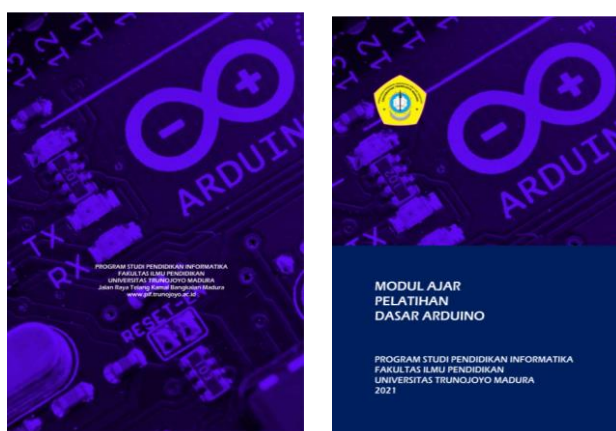
Adapted from (Chung & Lou, 2021)

### 3. RESULTS AND DISCUSSION

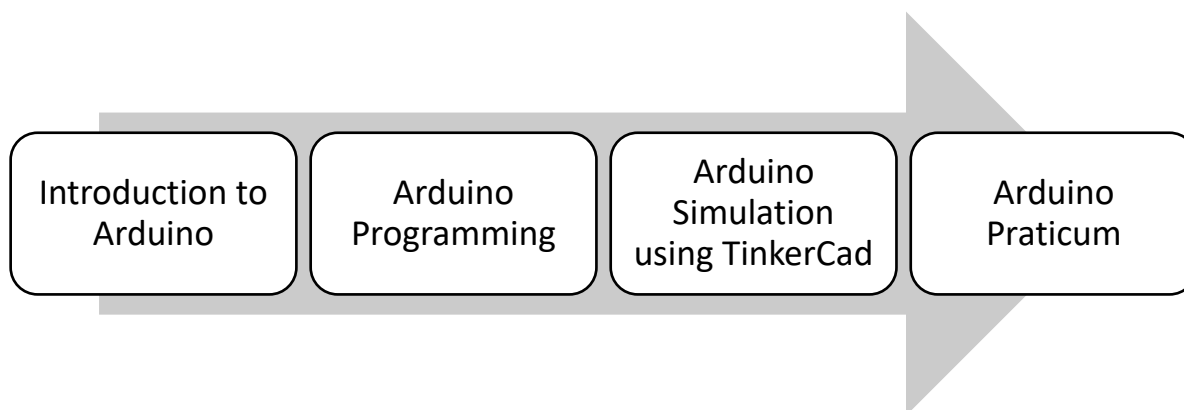
This section will explain the research results obtained until this Progress Report is compiled. It consists of Physical Computing-Based Teaching Modules and Arduino Basic Training Module Packages containing Arduino Uno and complementary Devices.

#### 3.1 Physical Computing Teaching Modul

The Teaching Module, as the cover page shows in Figure 2, is the result of this research. It collects materials studied by students. The materials included an introduction to Arduino Uno, Arduino Programming, Arduino Simulation using TinkerCad, and Practice using Arduino. Figure 3 shows the flow of material delivery in the Arduino Basic Training Teaching Module.



**Figure 2.** Physical Computing Teaching Modul



**Figure 3.** Stages of Material in the Arduino Basic Training Teaching Module

The following section describes the parts of the material collected in the Arduino Basic Training Teaching Module.

## Introduction to Arduino

The introductory Arduino material covered in this Teaching Module includes the following material:

- 1) Introduction to Microcontrollers;
- 2) Introduction to ATmega328;
- 3) Advantages of Arduino;
- 4) Types of Arduino;
- 5) Arduino Uno Components;
- 6) Arduino Usage Procedures.

## Arduino Programming

Arduino programming cannot be separated from the discussion about Arduino. This is because Arduino is a system that combines hardware and accompanying electronic devices and software in the form of programs that will later be embedded into Arduino.

Software or programs written using the Arduino programming language are called sketches. Sketches are written in a text editor called the Arduino IDE (Integrated Development Environment). The Arduino Programming Language is written similarly to the C Programming Language. The Arduino IDE can be downloaded openly on the page <https://www.arduino.cc/en/software> (**Figure 4**).



**Figure 4.** View of the [www.arduino.cc](https://www.arduino.cc)

Sketches are saved with the .ino file extension, which has features for cutting, copying, pasting, searching, or replacing text. The message area (console) provides feedback, saves and exports, and displays errors. The console displays the output text in the Arduino environment, including detailed error messages and other information. The lower right corner of the window displays the board and the serial port currently in use. The toolbar buttons (**Figure 5** and **Figure 6**) allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor. Arduino program code is usually called a sketch and is created using the C programming language. Programs or sketches written in the Arduino IDE can be directly compiled and uploaded to the Arduino Board.



Figure 5. Arduino Toolbar and Menu



Figure 6. Sketch in Arduino IDE

### Arduino Programming Structure

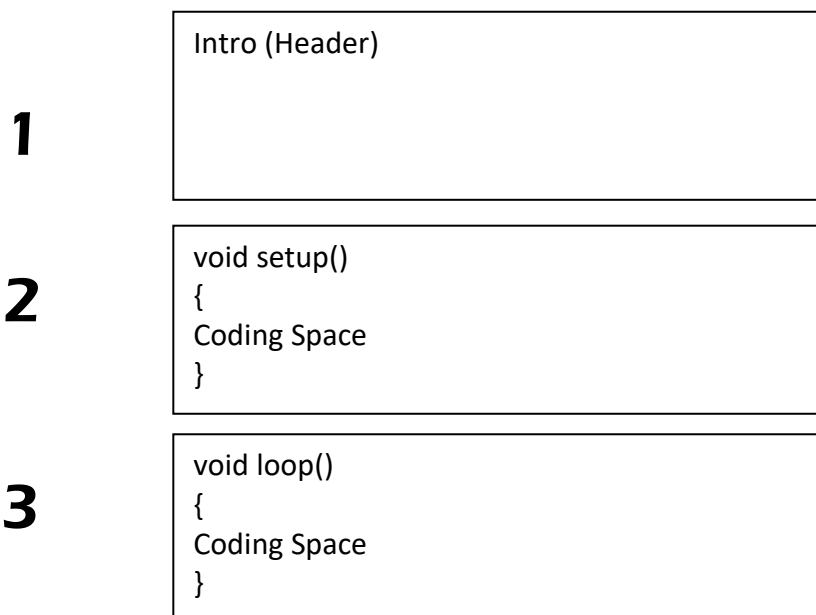


Figure 7. Arduino Sketches

In simple terms, sketches in Arduino are grouped into 3 blocks (**Figure 7**): Header, Setup, and Loop. For more complex programs, there will be other blocks in the form of supporting functions.

## Header

In this section, essential definitions are usually written that will be used later in the program, such as libraries and defining variables. The code in this block runs only once during compile time. Below is an example of code to declare the led variable (integer) and, at the same time, fill it with the number 13.

```
int led = 13;
```

## Setup

This is where the Arduino program starts, namely at the beginning or when the Arduino board is powered on. Usually, this block is filled with determining whether a pin is used as an input or output using the pinMode command. Variable initialization can also be done in this block.

```
// the setup routine runs once when you press reset:

void setup() { // initialize the digital pin as an output. pinMode(led, OUTPUT);
}

```

OUTPUT is a macro defined by Arduino, which means = 1. So, the command above is the same as pinMode (led, 1). A pin can be used as an OUTPUT or INPUT. If used as an output, it is ready to send electric current (a maximum of 100 mA) to the load it is connected to. If used as an INPUT, the pin has a high impedance and is ready to receive the current sent.

## Loop

This block will be executed continuously. When the program reaches the end of the block, it will continue by repeating the execution from the beginning of the block. The program will stop when the Arduino power button is turned off. This is where the primary function of our Arduino program is located.

```
void loop() {
digitalWrite(led, HIGH); // nyalakan LED delay(1000); // tunggu 1000 milidetik
digitalWrite(led, LOW); // matikan LED
delay(1000); //      tunggu 1000 milidetik
}

```



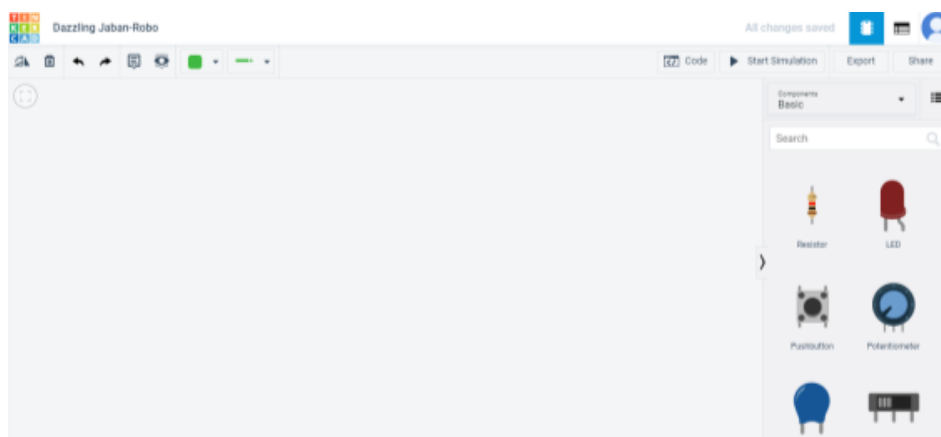
The `digitalWrite(pinNumber, value)` command instructs the Arduino to turn on or off the voltage at `pinNumber` depending on its value. So the command above `digitalWrite(led, HIGH)` will make pin number 13 (because in the header it is declared `led = 13`) have a voltage = 5V (HIGH). There are only two possible `digitalWrite` values, namely HIGH or LOW, which are actually integer values 1 or 0.

### Arduino Simulation using TinkerCad

TinkerCad is a web-based 3D Modeling Simulator Program that can be used by learners to learn Arduino without having an Arduino board and kit (**Figure 8**). It was first launched in Europe in 2011. This program was created by Kai Backman and his colleague Mikko Mononen. Currently, Tinkercad is managed by Autodesk. To use Tinkercad, access it on its website: [www.tinkercad.com](http://www.tinkercad.com) (**Figure 9**).



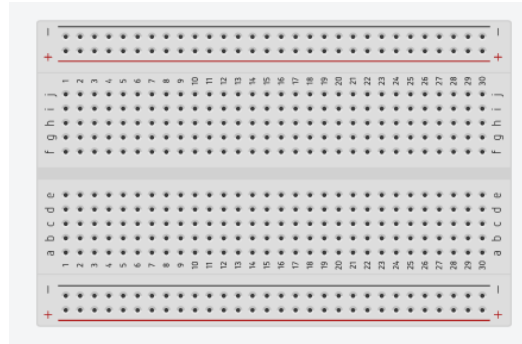
**Figure 8.** TinkerCad Logo



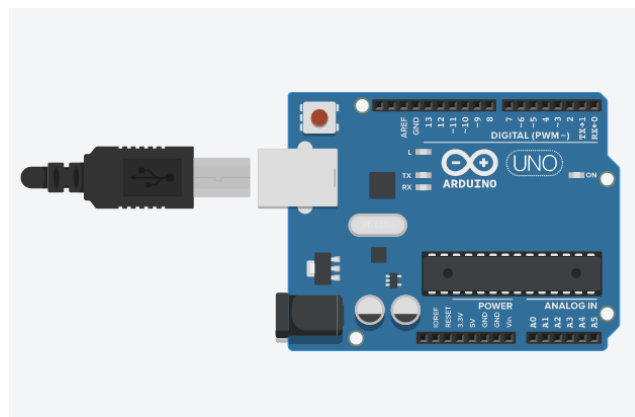
**Figure 9.** TinkerCad Circuits

### Arduino Simulation using TinkerCad

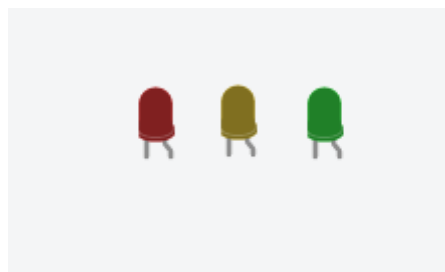
Arduino simulation using TinkerCad is very possible for beginners. Students can use Arduino as if they were using the Arduino board in physical form through the drag-and-drop activity of the devices available on the TinkerCad page. In the experiment written in the developed Teaching Module, materials used were a Breadboard (**Figure 10**), Arduino Uno, LED lights, and Resistors.



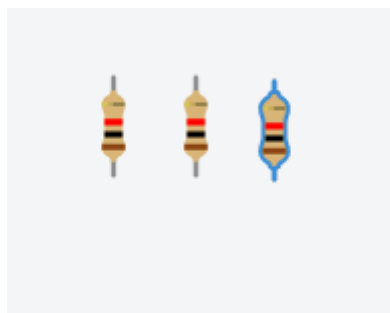
**Figure 10.** Breadboard in TinkerCad Simulator



**Figure 11.** Arduino Uno board in TinkerCad Simulator

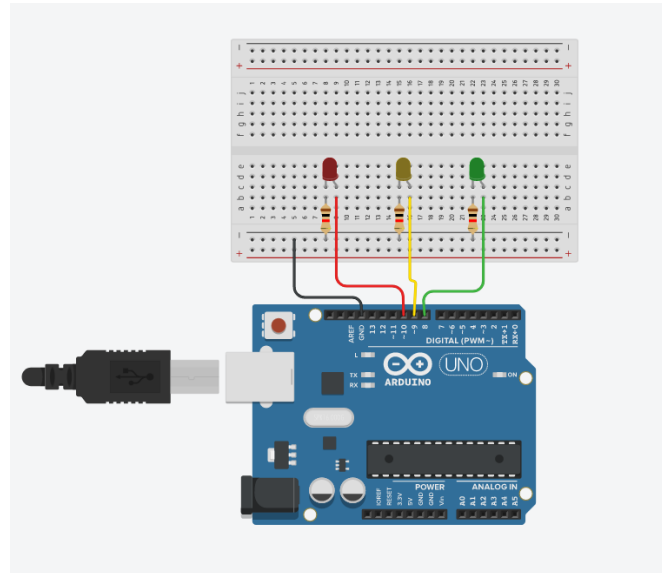


**Figure 12.** Led Lights in TinkerCad Simulator

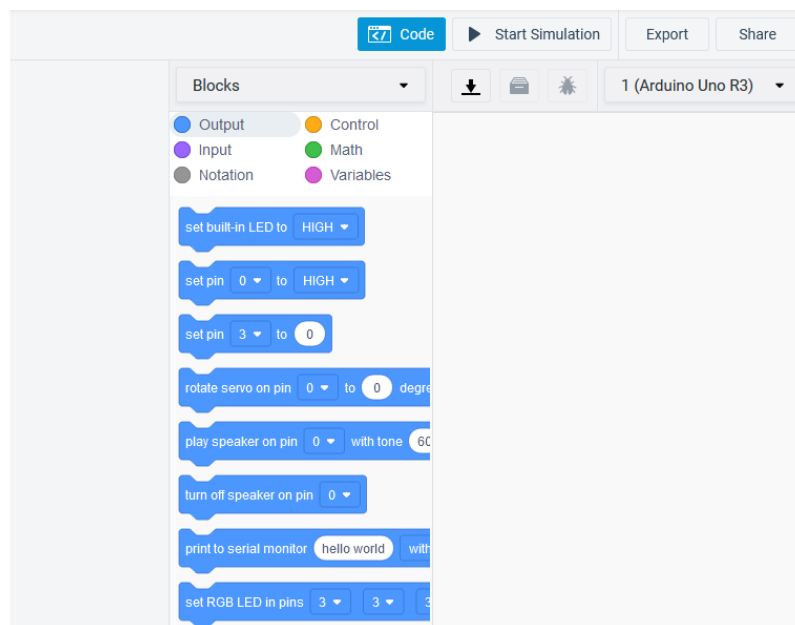


**Figure 13.** Resistors in the TinkerCad Simulator

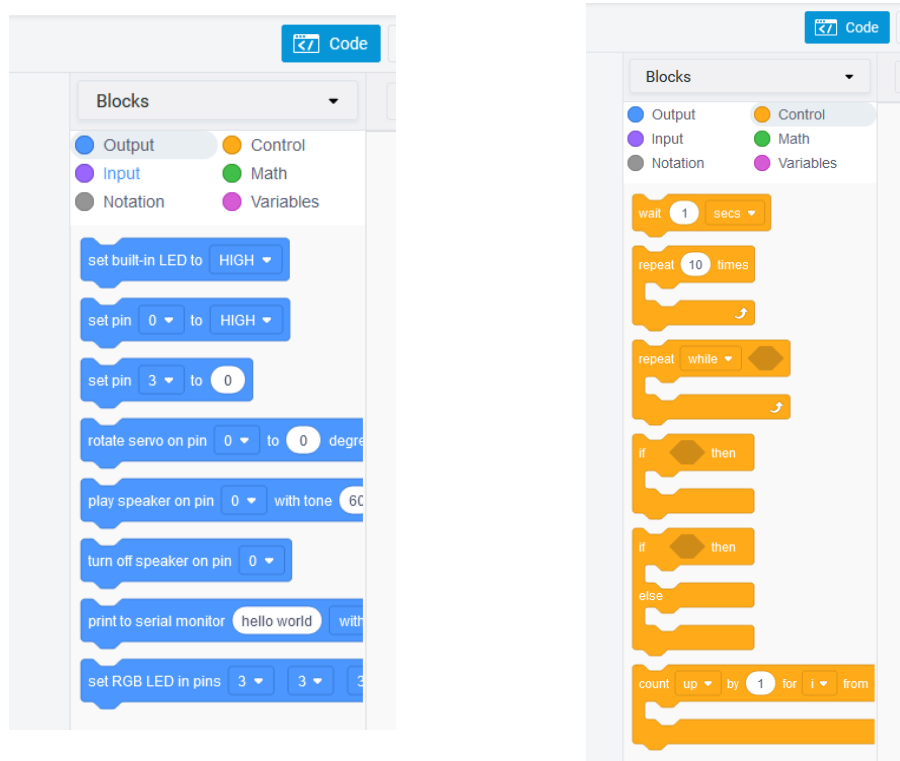
After students enter the TinkerCad simulator page, the next step is to ask them to create a series of components, as shown in **Figure 14**. The figure shows a series of components along with jumper cables. In the TinkerCad simulator, jumper cables can be formed after the components are installed.



**Figure 14.** Circuit in the TinkerCad Simulator

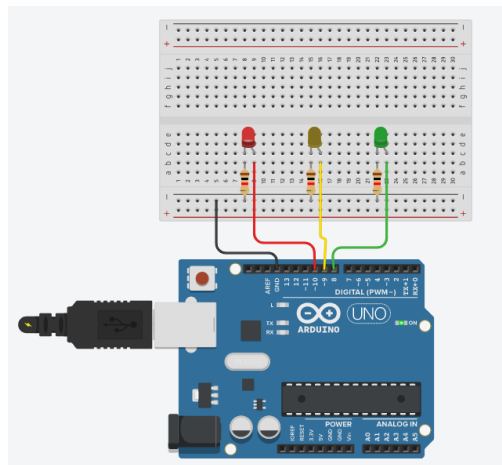


**Figure 15.** Block Code in TinkerCad Simulator

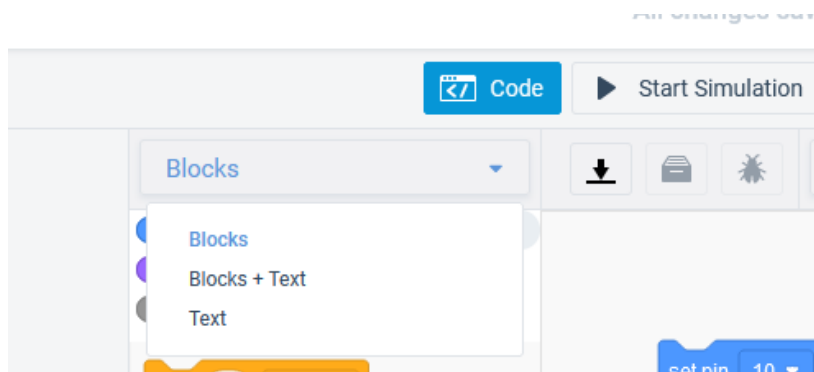


**Figure 16.** Program in Block Code in TinkerCad Simulator

**Figure 16** shows the program that has begun to be compiled in the Block Code in the TinkerCad Simulator. The block code has a command to turn on the LED lights alternately with a delay or pause for 1 second. **Figure 17** shows a circuit that can run commands as the code is compiled.

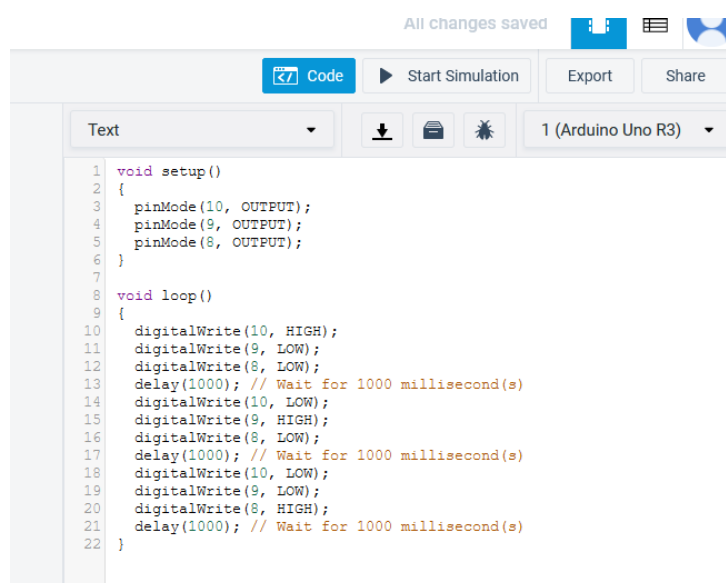


**Figure 17.** Series of components that have run the program



**Figure 18.** Text Program Starting to be Compiled

**Figure 18** shows the menu options in TinkerCad that allow you to write a program or sketch in text format. The simulations performed in the Teaching Module are also used to prepare text-based program codes. **Figure 19** shows the program code or sketch that has been prepared as text lines.



**Figure 19.** Program code structure that has been compiled from the TinkerCad Simulator

### Basic Arduino Training Module Package

This section explains the Basic Arduino Module Package, which is an integral part of the Teaching Module that has been explained previously. In one Basic Arduino Training Module package, several components are provided to carry out direct Practice using Arduino. One package consists of an Arduino Uno, BreadBoard, Resistor, LED Light, Jumper cable, and USB Cable. **Figure 20** shows an example of a packaged Basic Arduino Training Module package.



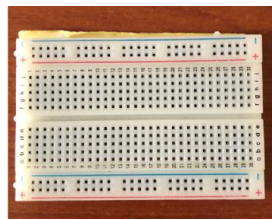
**Figure 20.** The Arduino Teaching Package

### Practice Using Arduino

In this section, the Teaching Module invites training participants or students to do direct Practice to develop a simple system using Arduino and the devices provided. First, students are set to learn the components, including Arduino Uno, BreadBoard, Resistors, LED lights, Jumper cables, and USB cables. **Figure 21** shows some of the components provided to practice using Arduino.



(a)



(b)



(c)



(d)



(e)

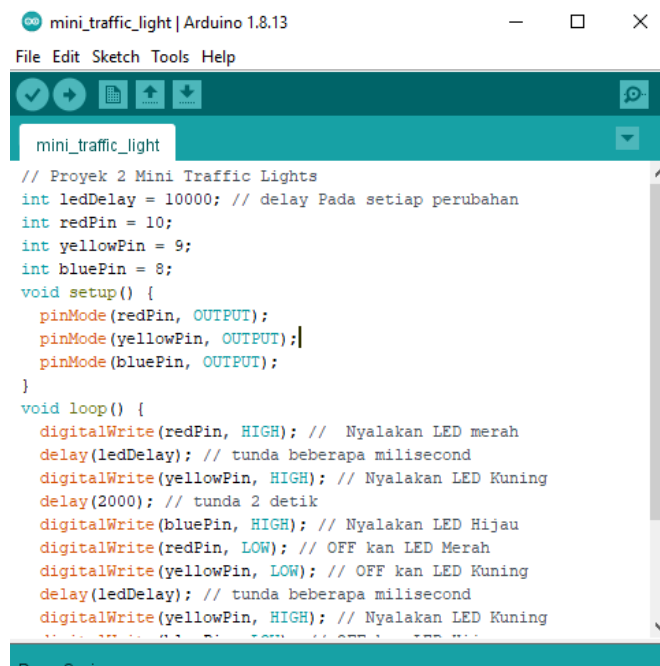
**Figure 21.** Components provided in the Basic Arduino Training Module Package: (a) Arduino Uno; (b) Breadboard; (c) Resistor; (d) LED Light; (e) USB Cable.

The next step is for students to install the Arduino IDE application (**Figure 22**). The Arduino IDE application, which can be downloaded for free from the Arduino page, is used to write program code. Next, students are asked to write program code. The Source Code can use the code above, as has been done in TinkerCad, or it can also be used as the program block below.



**Figure 22.** Arduino IDE download screen

```
// Proyek 2 Mini Traffic Lights
int ledDelay = 10000; // delay Pada setiap perubahan
int redPin = 10;
int yellowPin = 9;
int bluePin = 8;
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
void loop() {
  digitalWrite(redPin, HIGH); // Nyalakan LED merah
  delay(ledDelay); // tunda beberapa milisecond
  digitalWrite(yellowPin, HIGH); // Nyalakan LED Kuning
  delay(2000); // tunda 2 detik
  digitalWrite(bluePin, HIGH); // Nyalakan LED Hijau
  digitalWrite(redPin, LOW); // OFF kan LED Merah
  digitalWrite(yellowPin, LOW); // OFF kan LED Kuning
  delay(ledDelay); // tunda beberapa milisecond
  digitalWrite(yellowPin, HIGH); // Nyalakan LED Kuning
  digitalWrite(bluePin, LOW); // OFF kan LED Hijau
  delay(2000); // Tunda 2 detik
  digitalWrite(yellowPin, LOW); // OFF kan LED Kuning
  // Loop akan terus berulang
}
```



```

mini_traffic_light | Arduino 1.8.13
File Edit Sketch Tools Help
mini_traffic_light
// Proyek 2 Mini Traffic Lights
int ledDelay = 10000; // delay Pada setiap perubahan
int redPin = 10;
int yellowPin = 9;
int bluePin = 8;
void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
void loop() {
  digitalWrite(redPin, HIGH); // Nyalakan LED merah
  delay(ledDelay); // tunda beberapa milisecond
  digitalWrite(yellowPin, HIGH); // Nyalakan LED Kuning
  delay(2000); // tunda 2 detik
  digitalWrite(bluePin, HIGH); // Nyalakan LED Hijau
  digitalWrite(redPin, LOW); // OFF kan LED Merah
  digitalWrite(yellowPin, LOW); // OFF kan LED Kuning
  delay(ledDelay); // tunda beberapa milisecond
  digitalWrite(yellowPin, HIGH); // Nyalakan LED Kuning

```

**Figure 23.** Sketch view in Arduino IDE

#### 4. CONCLUSION

The systematics of the Physical Computing-based Teaching Module material for Informatics Learning produced from this study are structured material that supports the learning of beginner students who have never interacted with forms of Physical Computing. The Physical Computing used in this study is Arduino Uno. The systematics of the Physical Computing-based Teaching Module material for Informatics Learning developed in this study include Introduction to Arduino, Arduino Programming, Arduino Simulation using TinkerCad, and Practice using Arduino.

In addition to the Teaching Module, a product in the form of a Basic Arduino Training Module package is also produced. This Module Package is an accompaniment that is used together with the Teaching Module. One Module Package contains an Arduino Uno microcontroller, Breadboard, LED lights, USB cable, Jumper cable, resistor, and package.

#### ACKNOWLEDGMENT

The author would like to express his deepest gratitude to Universitas Trunojoyo Madura for the funding provided through the Research Scheme of Penelitian Pemula so that this research could be done. Thanks are also conveyed to SMK Al-Amin, Sumenep, for the opportunity to test the results of this research.

#### AUTHORS' NOTE

The authors declare that there is no conflict of interest regarding the publication of this article and confirm that the paper is free of plagiarism.



## REFERENCES

- Alden, D., & Tramonti, M. (2020). Computational Design Thinking and Physical Computing: Preliminary Observations of a Pilot Study. *Robotics*, 9(3). <https://doi.org/doi:10.3390/robotics9030071>
- Amalia, R., Assani, S., Effindi, M. A., Wijaya, E. Y., & Aini, N. (2023). Rancang Bangun Media Pembelajaran Algoritma Pemrograman Berbasis Android. *Jurnal Ilmiah Edutic*, 9(2). <https://doi.org/10.21107/edutic.v9i2.20215>
- Chung, C.-C., & Lou, S.-J. (2021). Physical Computing Strategy to Support Students' Coding Literacy: An Educational Experiment with Arduino Boards. *Applied Sciences*, 11. <https://doi.org/10.3390/app11041830>
- Effindi, M. A. (2020). *Informatics Learning: Pembelajaran Informatika bagi Pendidikan Dasar dan Menengah*. CV. Literasi Nusantara.
- Effindi, M. A. (2022). *Computational Thinking dalam Pembelajaran Informatika*. CV. Literasi Nusantara.
- Hodges, S., Sentance, S., Finney, J., & Ball, T. (2020). Physical computing: A Key Element of Modern Computer Science Education. *Computer*, 53(4). <https://doi.org/10.1109/MC.2019.2935058>
- Horn, M., & Bers, M. (2019). Tangible Computing. In *The Cambridge Handbook of Computing Education Research*. Cambridge University Press. <https://www.cambridge.org/core/books/abs/cambridge-handbook-of-computing-education-research/tangible-computing/3BE6A6D5D94493EF725C46896DACD46C>
- Jormanainen, I., & Tukiainen, M. (2020). *Attractive Educational Robotics Motivates Younger Students to Learn Programming and Computational Thinking*. TEEM'20: Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality. <https://doi.org/10.1145/3434780.3436676>
- Kartiansyah, A., Siswanto, H. W., Purnamasari, N., & Umanailo, M. C. B. (2021). Informatics As A Form Of Indonesia's Future Curriculum. *Psychology and Education*, 58(4), 1942–1950.
- Katterfeldt, E.-S., Cukurova, M., Spikol, D., & Cuartielles, D. (2018). Physical Computing with Plug-and-Play Toolkits: Key Recommendations for Collaborative Learning Implementations. *International Journal of Child-Computer Interaction*, 17, 72–82. <https://doi.org/10.1016/j.ijcci.2018.03.002>
- Roumen, G. J., & Fernaus, Y. (2021). Envisioning Arduino Action: A Collaborative Tool for Physical Computing in Educational Settings. *International Journal of Child-Computer Interaction*, 29. <https://doi.org/10.1016/j.ijcci.2021.100277>
- Vasilchenko, A., Venn-Wycherley, M., Dyson, M., & Abegao, F. R. (2020). Engaging Science and Engineering Students in Computing Education through Learner-Created Videos and Physical Computing Tools. *CEP '20: Proceedings of the 4th Conference on Computing Education Practice*. <https://dl.acm.org/doi/10.1145/3372356.3372372>

