The development of Auto-generated code for Programming PIC16F877 Microcontroller

Hilario A. Calinao Jr.[1], Nasher G. Jimenez[2], and Oliver R. Mariano[3]

[1-3]Bulacan State Univesit Brgy. Guinhawa City of Malolos, Bulacan, Philippines 3000

Abstract

Developing an embedded system using the PIC16F877 microcontroller requires a combination of software programming and hardware development skills, making it a complex undertaking. In the field of Electronics Engineering, Microprocessor Systems is a subject that specifically focuses on embedded system development. This study aimed to create the Auto-Generated Code for Programming PIC16F877 (AGCPIC) system, which encompasses both software and hardware components, to assist students in building embedded systems with the PIC16F877 microcontroller. The software component was developed using the Visual Basic 2010 programming platform and utilized a combination of textual and visual programming techniques for the microcontroller. To facilitate real-time response testing of the generated source code, an AGCPIC hardware development board was constructed. Thorough testing was conducted to ensure the reliability of the system, including the successful completion of unit testing (100% passed), integration testing (100% passed), and dependable systems testing. The overall testing results confirm the system's reliability. Moreover, based on the assessment summary, the AGCPIC system achieved an overall acceptability rating of 4.72, indicating a high level of acceptability. The statistical t-test results demonstrate the significance of the AGCPIC system's functionality, usability, efficiency, and subject matter content with a 95% confidence level.

Keywords: embedded system, microcontroller system, visual programming, IO devices

1 Introduction

The process of building an embedded system project is an essential requirement in the field of electronics engineering. Embedded systems involve a combination of hardware and software that enables the processing of input signals to perform specific tasks. In this course, students are tasked with creating programs that can be run on microcontrollers or microprocessors using low-level and high-level programming languages with cross-compilation techniques (CHED, 2018). Based on data gathered from electronics engineering students at Bulacan State University during the 2013-2014 school year, it was found that 76% of 124 students utilized microcontroller-based projects, with 60% of them using the PIC16F877 microcontroller. The data also revealed that 67.74% of students agreed that writing codes presented a challenge in developing an embedded system, while 29.84% agreed that testing the program source code and 29.03% agreed that interfacing with external components were challenging aspects.

Flowcharting knowledge plays a crucial role in the development of an embedded system project as it helps visualize the program algorithm's flow. Analyzing the flow of the program allows students to make necessary adjustments based on program requirements. Creating a program flowchart enables programmers to communicate their complex plans effectively before writing the actual program source code (Lehman, 2000). Neglecting the use of flowcharts can lead to disorganized program flow.

Additionally, it is important for students to possess the skills to create schematic diagrams before building and testing the program using actual hardware (Bayliss & Hardy, 2012). In the process of building an embedded system program, students must have the knowledge to create flowcharts, convert them into hardcode programs, compile the program for deployment on actual hardware, create schematic diagrams for the system, interface input and output components, and validate the codes based on actual hardware responses. These skills are crucial for students to successfully complete the electronics engineering course.

When combined with text-based programming, visual programming techniques simplify and enhance the development of embedded system programming and improve 53 students' computational thinking abilities (Wu & Su, 2021). A study conducted on a group of students in Jordan demonstrated an improvement in performance and attitude towards programming when using 3D visual programming tools (Al-Tahat, 2019). Software tools such as MATLAB and LabView utilize graphical programming and provide a high-level programming language with graphical matrix and environment for data analysis, computation, and visualization during programming and simulation. Another software, Flowcode, developed by Matrix Multimedia Inc., uses flowchart symbols to create program source code. While this study aims to develop a visual programming software specifically for students working on embedded system projects, it focuses on simplifying programming by creating flowcharts, aiding students in visualizing the interface between specific input and output devices, and validating program responses using a hardware system.

Overall, this study aims to develop visual programming software that will facilitate the process of building embedded system projects for students. It will simplify programming tasks by allowing the creation of flowcharts to represent program flow, aid in visualizing device interface configurations, and validate program responses using real-world scenarios with the help of hardware components.

2 Materials and Methods

This study utilized a combined descriptive and applied research which uses surveys and interviews to gather information from the respondents and it also applies the scientific knowledge as a solution to the problem. This study develops a new system and methods to solve the challenges in building an embedded system project using PIC16F877 microcontroller. To ascertain the requirements of this study, the respondents are required to provide answers to the following questions: (1) Have you utilized a microcontroller-based project? (2) Which type of microcontroller did you use? (3) What were the most challenging phases you encountered during the construction of your project?

The development process of auto-generated code for programming the PIC16F877 microcontroller involves both software and hardware components. In this study, the Visual Basic 10 Programming language is used to create a graphical user interface (GUI) for the software. The software component has a hierarchical structure depicted in Figure 1, which consists of four main options: file, tools, databases, and instantiate. Additionally, there is an additional help function available. Each of these options includes sub-options that provide the necessary functions for developing the source code and schematic diagram of the embedded system using the software.

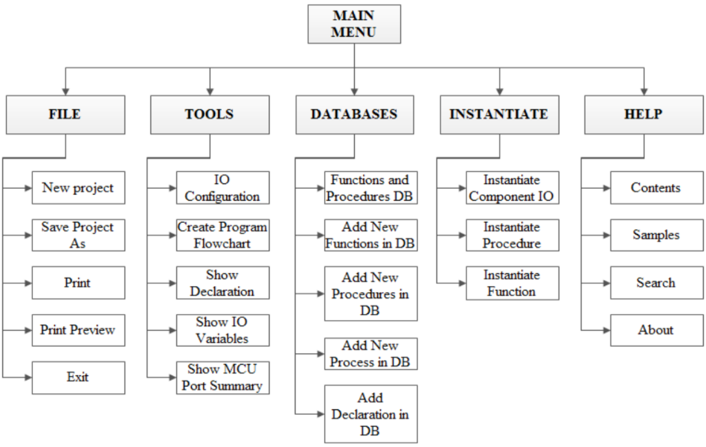

Figure 1. Hierarchical structure of the AGCPIC software

AGCPIC software. The graphical user interface (GUI) of the AGCPIC software Figure 2. The software shows the optional tab for schematic diagram, generated code tab and all files tab. It has an option to display the flowchart created by the user when programming using the software. The schematic diagram provided by the software can be used by the user to visualize the hardware equivalent of embedded system project. The sample schematic diagram in programming LED matrix with pushbutton is shown in Figure 2 below.

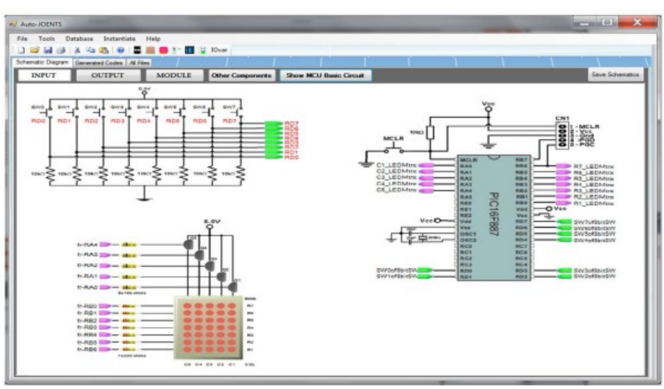

Figure 2. Sample schematic diagrams using the AGCPIC software.

The software allows users to create flowcharts and schematic diagrams, which can be exported as image files. Each symbol used in the flowchart corresponds to a programming code. The software supports dataflow, input and output configuration, device properties editing, and schematic diagram editing, all of which can be translated into code. Once the microcontroller is visually programmed, users have the option to generate the source code as a text document for testing purposes. The flowchart creation process involves dragging and dropping symbols, which are color-coded to represent their functions and labeled with their corresponding programming code. An example programming flowchart is depicted in Figure 3. However, the source code cannot be directly used by the hardware system. It needs to be compiled using cross-compiler software like MikroBASIC to generate the machine language file (HEX file). Finally, the Microchip PICkit2 hardware downloader is used to burn or upload the compiled source code into the actual hardware system for testing and evaluating the program's real-world response.

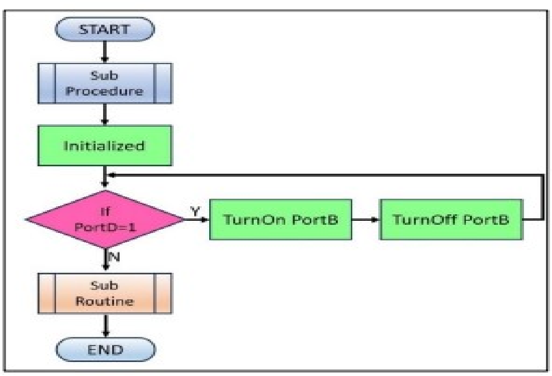

Figure 3. Sample Program Flowchart Using AGCPIC Software

AGCPIC hardware. The block diagram in Figure 4 illustrates the hardware components and parts of the AGCPIC development system board. The development board can be powered by various power sources and features different input-output devices for simulating program responses. It also includes a small breadboard for prototyping purposes and provides communication ports and Arduino slots for hardware expansion.

To test and simulate the source code generated using the AGCPIC software, a specialized development board called the Development Board for PIC16F877 is utilized. This board is specifically designed to work with the AGCPIC software, mirroring the input and output devices present in the software. Figure 5 showcases the physical hardware that can be used to verify the accuracy of the generated code.

Developing an embedded system using the AGCPIC system entails several requirements. Users must possess a fundamental understanding of the AGCPIC software, MikroBASIC compiler, and PICKit2 hardware downloader. Additionally, knowledge of the PIC16F877 microcontroller and various input-output (IO) devices is necessary to fulfill both software and hardware prerequisites. Proficiency in creating schematic diagrams and flowcharting is also essential.

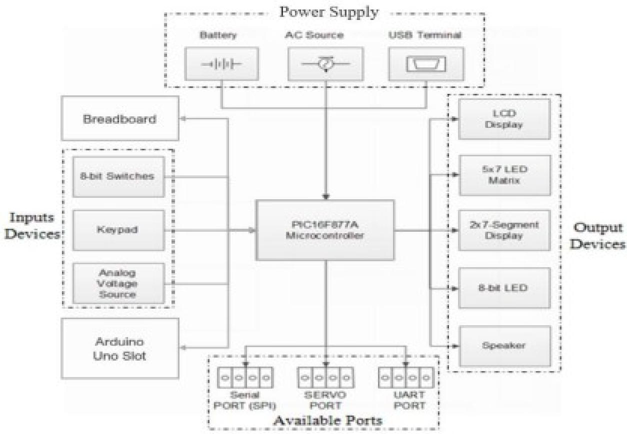

Figure 4. Block diagram of AGCPIC hardware development board

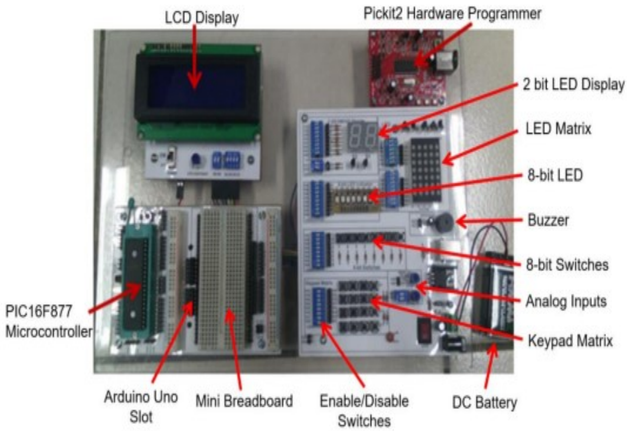

Figure 5. Actual hardware development board

When undertaking an embedded system project, users must follow specific steps to apply basic programming methods. These steps include: 1) Defining the problem, 2) Creating a schematic diagram, 3) Developing a flowchart, and 4) Conducting project testing (Alciatore, 2018). Figure 6 presents a comprehensive illustration of the processes involved in programming using the AGCPIC software and hardware development. The output of this system encompasses the schematic diagram, flowchart, and source code of the embedded system. Furthermore, the physical hardware development board enables the observation of the actual system response.

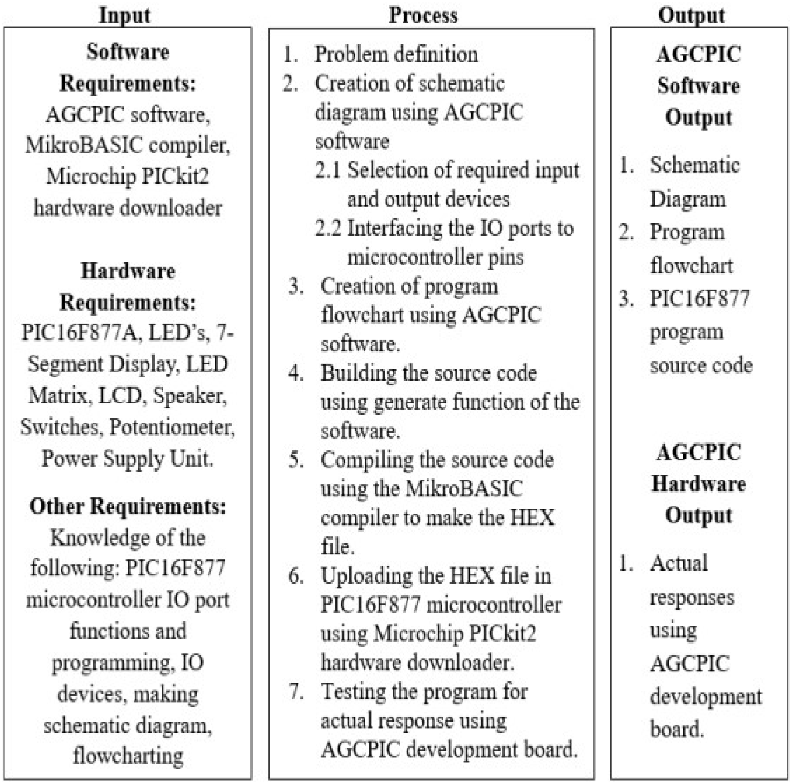

Figure 6. Input-process-output diagram of using the AGCPIC system.

Reliability of AGCPIC. Testing at various levels is essential in assessing the system's reliability. The reliability of AGCPIC refers to its ability to execute its anticipated functions with no error (ISO/IEC, 2001). The system underwent unit testing to ensure the functionality of individual components. Successful completion led to integration testing, where these components were combined with others. The systems testing was also conducted to check the overall reliability of system.

Acceptability of AGCPIC. The acceptability of the AGCPIC system was determined by 149 conducting surveys on the students, faculty, and industry personnel. And it was conducted after 150 the AGCPIC software and hardware system was completed. In the school year 2015-2016, 70 151 fifth-year electronics engineering students participated in the survey.

There are also 8 faculty 152 members and 2 industry personnel who have knowledge of programming who participated to 153 determine the acceptability of the AGCPIC system. Table 1 shows the tabulated respondents 154 of the study. The evaluation is used to check the acceptability of the system through its 155 functionality, usability, efficiency, and subject matter content of the AGCPIC system.

Table 1. Respondents of the study

Respondents	Frequency	Percentage
Students	70	87.5%
Faculty members	8	10%
Industry personnel	2	2.5%
TOTAL	**80**	**100%**

In this study, the user perspective was determined using the Likert scale, consisting of several statements. Each statement had five response alternatives, ranging from 5) Highly Acceptable to 1) Highly Unacceptable. To consolidate the collective response into a single value, the central tendency, such as the mean or average, was employed. The weighted mean formula was utilized to represent the individual measurements of each respondent as a single number. The statements were evaluated on a scale from 5 to 1, based on the perception of the respondents. The results were then calculated to determine the weighted mean within the specified range, and the interpretation can be found in Table 2.

Table 2. Likert scale

Scale	Range	Verbal Interpretation
5	4.51 – 5.0	Highly Acceptable
4	3.51 – 4.5	Moderately Acceptable
3	2.51 – 3.5	Neutral
2	1.51 – 2.5	Moderately Unacceptable
1	1.0 – 1.5	Highly Unacceptable

To further assess the acceptability of the study, a one-sample t-test is utilized. This statistical test allows us to examine the gathered samples in relation to a specific population (Al-kassab, 2022). The null hypothesis (H0) is examined using a one-sample t-test with a 95% confidence level, corresponding & Fagerland, 2015). In this study, the null hypotheses (Ho) are as follows:

Ho1: The functionality of the AGCPIC system does not significantly contribute to the development of an embedded system project.

Ho2: The usability of the AGCPIC system does not significantly contribute to the development of an embedded system project.

Ho3: The efficiency of the AGCPIC system does not significantly contribute to the development of an embedded system project.

Ho4: The subject matter of the AGCPIC system does not significantly contribute to the development of an embedded system project.

3 Results and Discussion

This study aims to develop an auto-generated code for programming the PIC16F877 microcontroller. The software used Visual Basic 2010 as the programming platform for creating this software. The study consists of both software and hardware components, enabling users to create programs using a combination of textual and visual programming techniques. To test the programs created using the software, users need to upload them to the development hardware. The software employs the Beginners All-purpose Symbolic Instruction Code (BASIC), specifically designed for PIC microcontrollers, to generate the source code for users' embedded system projects. This source code is then transferred to a cross-compiler, the MikroBASIC programming software, which generates the machine language or HEX file required by the microcontroller. The Pickit2 hardware downloading software is used to upload the compiled source code to the microcontroller.

To observe the actual response of the user-created program, the AGCPIC development board is used. The study's results, including unit testing and integration testing, can be found in Table 3. The testing results demonstrate the reliability of the individual components and functions of the AGCPIC development board, affirming its suitability for embedded system project development.

Table 3. Results of unit testing and the integration testing of the development board.

Table 2. Results of unit testing and the integration testing of the development board.

Type of Testing	Function	IO Devices	Result
Unit Test	Sends High Signal to MCU	SW0	Passed
Unit Test	Sends High Signal to MCU	SW1	Passed
Unit Test	Sends High Signal to MCU	SW2	Passed
Unit Test	Sends High Signal to MCU	SW3	Passed
Unit Test	Sends High Signal to MCU	SW4	Passed
Unit Test	Sends High Signal to MCU	SW5	Passed
Unit Test	Sends High Signal to MCU	SW6	Passed
Unit Test	Sends High Signal to MCU	SW7	Passed
Unit Test	Respond to Pressed Key	Keypad Matrix	Passed
Unit Test	Generates 0-5Volts	Analog source 1 (VR1)	Passed
Unit Test	Generates 0-5Volts	Analog source 2 (VR2)	Passed
Unit Test	Blinking	LED0	Passed
Unit Test	Blinking	LED1	Passed
Unit Test	Blinking	LED2	Passed
Unit Test	Blinking	LED3	Passed
Unit Test	Blinking	LED4	Passed
Unit Test	Blinking	LED5	Passed
Unit Test	Blinking	LED6	Passed
Unit Test	Blinking	LED7	Passed
Unit Test	Blinking	LED0-7	Passed
Unit Test	Running Light (Right to Left)	LED0-7	Passed
Unit Test	Running Light (Left to Right)	LED0-7	Passed
Unit Test	Display numbers	7-Segment One's	Passed
Unit Test	Display numbers	7-Segment Ten's	Passed
Unit Test	Counting	7-Segment One's and Ten's	Passed
Unit Test	Display numbers	LED Matrix	Passed
Unit Test	Display Alphabet	LED Matrix	Passed
Unit Test	Counting	LED Matrix	Passed
Unit Test	Display numbers	LCD Display	Passed
Unit Test	Display Alphabet	LCD Display	Passed
Unit Test	Generate Sound	SPEAKER	Passed
Integration Test	Communicate Port A to IO devices	7-Segment Display	Passed
Integration Test	Communicate Port A to IO devices	LED Matrix	Passed
Integration Test	Communicate Port A to IO devices	Analog Input	Passed
Integration Test	Communicate Port B to IO devices	7-Segment Display	Passed
Integration Test	Communicate Port B to IO devices	8-Bit LED Indicator	Passed
Integration Test	Communicate Port B to IO devices	LED Matrix	Passed
Integration Test	Communicate Port C to IO devices	7-Segment Display	Passed
Integration Test	Communicate Port C to IO devices	8-Bit LED Indicator	Passed
Integration Test	Communicate Port C to IO devices	LED Matrix	Passed
Integration Test	Communicate Port C to IO devices	Speaker	Passed
Integration Test	Communicate Port D to IO devices	8-Bit Switches	Passed
Integration Test	Communicate Port D to IO devices	Keypad Matrix	Passed
Integration Test	Communicate Port D to IO devices	Speaker	Passed
Integration Test	Communicate Port E to IO devices	Analog Input	Passed
Integration Test	Communicate Port E to IO devices	8-Bit LED Indicator	Passed
Integration Test	Communicate Port E to IO devices	Speaker	Passed

The system testing follows the processes listed on the figure 6 when using the AGCPIC system. The sample schematic diagram and program flowchart made by using the AGCPIC system graphical user interface (GUI) is shown in figure 7.

In Table 4, the descriptive measures of the acceptability level of the AGCPIC system are presented, specifically focusing on its functionality. The evaluation of the system's functionality encompasses various statements that evaluate its capacity to fulfill the user's implicit requirements.

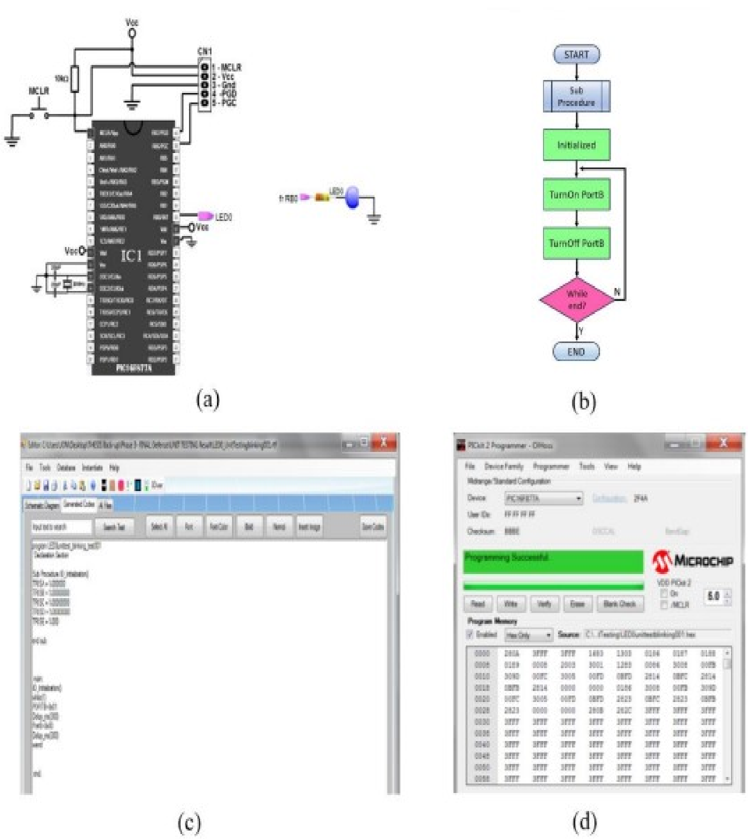

Figure 7. Systems testing results for AGCPIC: (a) Sample schematic diagram using the software; (b) Sample flowchart made using the software; (c) Sample generated source code were successfully created using

the software; (d) The created machine language was successfully burned in the PIC16F877 microcontroller using MicroChip PICkit2 hardware programmer.

Items		Frequency					Mean	Verbal Interpretation
		5	4	3	2	1		
1	It is easy to develop an embedded system project when the schematic diagram is made using the software.	57	21	2	0	0	4.69	Highly Acceptable
2	The schematic diagram made using AGCPIC software for the project is useful	55	22	3	0	0	4.65	Highly Acceptable
3	The AGCPIC software automatically generates codes that are functional based on user demand	58	18	4	0	0	4.68	Highly Acceptable
4	The schematic diagram generated by the software satisfies the user	57	21	2	0	0	4.69	Highly Acceptable
5	The flowchart made using the AGCPIC software met the users' requirements	64	15	1	0	0	4.79	Highly Acceptable
6	It is easy to develop a system using AGCPIC software where each block is equivalent to process codes	51	24	5	0	0	4.58	Highly Acceptable
7	The generated codes by the software satisfy the requirements of the user	59	21	0	0	0	4.74	Highly Acceptable
8	The schematic diagram created using the software complements the users' requirement	63	16	1	0	0	4.78	Highly Acceptable
9	AGCPIC software guides the programmer on a more realistic model of the embedded system project	70	10	0	0	0	4.88	Highly Acceptable
10	It is easy to develop an embedded system when codes are available for basic functionality like communications between devices using the AGCPIC software	52	23	5	0	0	4.59	Highly Acceptable
11	I am satisfied with AGCPIC software based on the functions available on the software.	72	8	0	0	0	4.90	Highly Acceptable
Average mean					**4.72**			**Highly Acceptable**

Table 5 presents the evaluation of the system's usability comprises a set of statements 212 aimed at assessing the system's capacity to be comprehended, learned, and operated effectively, 213 as well as its overall appeal to the user.

Table 5. Descriptive measures of the level of acceptability of the AGCPIC in terms of usability

Items		Frequency 5	4	3	2	1	Mean	Verbal Interpretation
1	The software provides a graphical editor that supports the modeling of flowcharts useful to the user	65	14	1	0	0	4.80	Highly Acceptable
2	The software provides a graphical editor that supports the modeling of schematic diagrams useful to the user	59	19	2	0	0	4.71	Highly Acceptable
3	The input and output ports are easy to configure	68	12	0	0	0	4.85	Highly Acceptable
4	The schematic diagrams are easy to interface with the microcontrollers	69	9	2	0	0	4.84	Highly Acceptable
5	All screens share a common organization and style to create familiarity for users.	59	21	0	0	0	4.74	Highly Acceptable
6	Flowcharting using the AGCPIC software is easy	65	13	2	0	0	4.79	Highly Acceptable
7	The schematic diagram, flowchart, and source code are easily accessible and understood	59	19	2	0	0	4.71	Highly Acceptable
8	Input devices are easy to configure	60	20	0	0	0	4.75	Highly Acceptable
9	Output devices are easy to configure	64	14	2	0	0	4.78	Highly Acceptable
10	The basic schematic diagram is easy to understand	63	15	2	0	0	4.76	Highly Acceptable
11	The visualization provides visual clues for spontaneous navigation.	58	18	4	0	0	4.68	Highly Acceptable
12	The codes generated using the AGCPIC software are understandable	61	18	1	0	0	4.75	Highly Acceptable
13	Flowcharts are easily understood using the AGCPIC software	61	17	2	0	0	4.74	Highly Acceptable
14	The data flow is easy to understand using the software	58	21	1	0	0	4.71	Highly Acceptable
Average mean							**4.76**	**Highly Acceptable**

Table 6 displays the evaluation of the respondents regarding the efficiency of the AGCPIC system. The efficiency assessment of the system consists of various statements that assess its ability to deliver appropriate response times, processing times, and resource utilization when the software is utilized.

Items		Frequency					Mean	Verbal Interpretation
		5	4	3	2	1		
1	AGCPIC software makes the user more productive	56	22	2	0	0	4.68	Highly Acceptable
2	AGCPIC software gives the user more time to think about the development of the system rather than thinking about the validity of the syntaxes for each module	60	19	1	0	0	4.74	Highly Acceptable
3	Using the AGCPIC system, the user does not need an external programmer to build an embedded system project	67	11	2	0	0	4.81	Highly Acceptable
4	AGCPIC software reduces the amount of time to develop an embedded system or MCU-based project	57	23	0	0	0	4.71	Highly Acceptable
Average mean		**4.73**						**Highly Acceptable**

The evaluation of the subject matter pertaining to the AGCPIC system is presented in Table 7. This evaluation involves multiple statements aimed at assessing the system's ability to enhance knowledge, understanding of concepts, and skills related to the subject of microprocessor systems. Additionally, the subject matter evaluation will determine whether the system is perceived as beneficial in the context of the microprocessor system subject, based on the feedback provided by the respondents.

Items		Frequency					Mean	Verbal Interpretation
		5	4	3	2	1		
1	The AGCPIC flowchart gives me an idea of how the program flow	58	22	0	0	0	4.73	Highly Acceptable
2	AGCPIC programming is helpful in understanding the microcontroller systems and design	62	14	4	0	0	4.73	Highly Acceptable
3	The circuit diagram made using the AGCPIC is important in creating an embedded system	60	18	2	0	0	4.73	Highly Acceptable
4	The AGCPIC software contributes to my knowledge of microcontrollers and interfacing with other I/O devices, memory, and other systems	56	21	3	0	0	4.66	Highly Acceptable
5	The AGCPIC software contributes to my understanding of the basic components needed on microprocessors and microcontroller system subject	55	19	6	0	0	4.61	Highly Acceptable
6	The software contributes to my knowledge of building an embedded system with a PIC16F877 microcontroller	56	20	4	0	0	4.65	Highly Acceptable
7	AGCPIC system significantly contributes to my skills in building my MCU-based project	55	20	5	0	0	4.63	Highly Acceptable
8	The software contributes to my understanding of the concept behind microprocessors and microcontrollers subject	50	27	3	0	0	4.59	Highly Acceptable
Average mean		**4.66**						**Highly Acceptable**

Table 8 presents a summary assessment of the acceptability of the auto-generated code 228 for programming PIC16F877A. The acceptability assessment encompasses four dimensions: 229 functionality, usability, efficiency, and subject matter content of the system.

Table 8. Assessment summary for acceptability of AGCPIC system

Variables	Mean	Verbal Interpretation
Functionality	4.72	Highly Acceptable
Usability	4.76	Highly Acceptable
Efficiency	4.66	Highly Acceptable
Subject Matter Content	4.66	Highly Acceptable
OVERALL MEAN	**4.72**	**Highly Acceptable**

The overall mean score for acceptability is 4.72, indicating a ”highly acceptable” rating. The evaluation demonstrates that the AGCPIC system successfully meets the implied needs of the user.

The one sample t-test result to evaluate the hypothesis of the study are shown in table 9 and table 10. These results indicate that for all variables (Functionality, Usability, Efficiency, and Subject Matter), the t-values are significantly higher than the test value of 3.5. The p-values are all 0.000, which is less than the typical significance level of 0.05, suggesting strong evidence to reject the null hypothesis. The mean differences are all positive, indicating that the mean scores for each variable are significantly higher than the test value. The 95% confidence intervals of the difference for each variable do not include the test value, further supporting the conclusion of statistical significance of the test variables. Overall, these results suggest that the respondents' perceptions of functionality, usability, efficiency, and subject matter are statistically significant in the development of an embedded system project.

Table 9. One-Sample Statistics

	N	Mean	Std. Deviation	Std. Error Mean
Functionality	80	4.7216	0.3140	0.0351
Usability	80	4.7571	0.3149	0.0352
Efficiency	80	4.7344	0.3498	0.0391
Subject Matter	80	4.6641	0.4064	0.0454

Table 10. One-Sample Test

	Test Value = 3.5					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
Functionality	34.793	79	0.0000	1.22159	1.1517	1.2915
Usability	35.710	79	0.0000	1.25714	1.1871	1.3272
Efficiency	31.560	79	0.0000	1.23438	1.1565	1.3122
Subject Matter	25.617	79	0.0000	1.16406	1.0736	1.2545

The AGCPIC testing and evaluation results proves that it is comprehensible, easy to learn, operate, and beneficial for the user. Furthermore, when applied in developing an embedded system project, the AGCPIC system exhibits appropriate response times, processing efficiency, and optimal resource utilization. The evaluation also highlights the system's effectiveness in aiding the development of embedded system projects and contributing to the understanding of microprocessor system subjects. Below are the shared experiences and impressions of the students who utilized the AGCPIC system for programming the PIC16F877 microcontroller.

"It will provide a way for students to understand the microcontroller-microprocessor design easily without having a hard time analyzing and imagining every step" – R.A.P. Agda
"The programming method is more interactive and is easier to debug" -R.L. Intal
"It is much easier to know what went wrong" – R. Evanghelista
"More concepts-ideas in inventing device rather than focusing on the codes" – K. Gojo
"It will help the programmer to visualize the progress of the project" – L. Bastez
"It boosts interest of the student to know more on how to program an embedded system" – K. Falcutila

4 Conclusion

The primary goal of this study is achieved through the creation of autogenerated code for programming the PIC16F877 microcontroller, encompassing both the hardware and software components of the system. The development of the AGCPIC system simplifies the programming and testing processes involved in working with embedded systems. The study encompasses various tests aimed at assessing the reliability of the system. These tests include unit testing, which verifies the correct functioning of each individual component of the project. Integration testing is conducted to ensure the continued reliability of the hardware once all the components are assembled. Additionally, the project undergoes systems testing to assess the overall reliability of the system, encompassing both the software and hardware aspects. The software-generated program is subjected to validation, wherein the AGCPIC development board is utilized to observe the real-time response of the student's embedded system project. It is noteworthy that all the test scenarios conducted in this study to evaluate reliability were successfully passed. The system assesses the acceptability level of its functionality, usability, efficiency, and subject matter content, all of which are determined to have a score of 4.72 which is highly acceptable. The statistical test results shows that the functionality, usability, efficiency, and subject matter content are significant in the development of embedded system project with 95% confidence level. Additionally, several students have provided positive feedback after utilizing the system for embedded system programming.

5 Acknowledgement

The authors would like to acknowledge the ECE students, ECE faculty, industry personnel, and support from Bulacan State University.

References

[1] Alciatore, D. (2018). Loose Leaf for Introduction to Mechatronics and Measurement Systems (5th edition). McGraw Hill.

[2] Al-kassab, M. (2022). The Use of One Sample t-Test in the Real Data. JOURNAL OF ADVANCES IN MATHEMATICS, 21, 134–138. https://doi.org/10.24297/jam.v21i.9279

[3] Al-Tahat, K. (2019). The Impact of a 3D Visual Programming Tool on Students' Performance and Attitude in Computer Programming: A Case Study in Jordan. Journal of Cases on Information Technology (JCIT), 21(1), 52–64. https://doi.org/10.4018/JCIT.2019010104

[4] Bayliss, C. R., & Hardy, B. J. (2012). Transmission and Distribution Electrical Engineering.

[5] CHED. (2018). CHED. CHED. https://ched.gov.ph/

[6] ISO/IEC. (2001). ISO/IEC 9126-1:2001. ISO.

[7] https://www.iso.org/cms/render/live/en/sites/isoorg/contents /data/standard/02/27/22749.html

[8] Laake, P., & Fagerland, M. W. (2015). Chapter 11—Statistical Inference. In P. Laake, H. B. Benestad, & B. R. Olsen (Eds.), Research in Medical and Biological Sciences (Second Edition) (pp. 379–430). Academic Press. https://doi.org/10.1016/B978-0-12-799943- 308 2.00011-2

[9] Lehman, M. (2000). Flowcharting Made Simple. Undefined. https://www.semanticscholar.org/paper/Flowcharting-Made-Simple-Lehman/71fbc79044c41add234c6bf0d98162cae3eaae27

[10] Wu, S.-Y., & Su, Y.-S. (2021). Visual Programming Environments and Computational Thinking Performance of Fifth- and Sixth-Grade Students. Journal of Educational Computing Research, 59(6), 1075–1092. https://doi.org/10.1177/0735633120988807