Paper ID #6388

Modern Embedded Systems as a Platform for Problem Solving in Freshman Engineering: What is the Best Option?

Mr. John W Pritchard, Iowa State University
Dr. Mani Mina, Iowa State University

Modern Embedded Systems as a Platform for Problem Solving: What is the Best Option?

John Pritchard[1] and Mani Mina[1]
[1]Electrical and Computer Engineering, Iowa State University, Ames, IA 50011, USA

I. Introduction

Students and hobbyists today are met with a plethora of electronics projects that can be easily completed with the wide variety of online resources and extensive documentation. Many of these projects include the use of high level embedded systems that serve as a “black box” for electronic control of sensors, actuators, motors, wireless communication, and other complex systems [1-6]. Recently, a trend has emerged in which these development platforms have become smaller, easier to use, open source, and affordable. This trend has enabled interesting projects that aim to introduce new technologies, inspire technological direction, provide capabilities to the underprivileged, and also to educate. In particular, many of these development platforms have made their way to the classroom, especially for early engineering education with the focus of problem solving [7-11]. However, there are many different systems to choose from with a variety of capabilities from an assortment of vendors, and some may or may not be suitable for educational purposes. Great efforts have been made to study different embedded systems [12-14], but these studies are generally created for a specific audience and do not differentiate between the many available systems on the market. This work attempts to bring an evaluation method, which differentiates different embedded platforms and is applicable to a broad audience, ranging from electronics enthusiasts to university instructors.

After investigation and discussion with different educators and users, we propose four parameters characteristic of the different embedded systems for this work:

1. Hardware-intensive (HI)
2. Software-intensive (SI)
3. Ease-of-implementation (EI)
4. Course/application relevance (CAR)

Hardware-intensive (HI) refers to the level at which a user is required to focus on hardware challenges. Similarly, software-intensive (SI) refers to the level at which a user is required to focus on software challenges. Since there are many different platforms that offer many differing options and capabilities, a single embedded system can be configured to provide for many different needs. A primary goal of this work is to provide a first-level evaluation method that may determine which systems can fit general needs right out of the box. Platforms that are easy to implement are those that are adaptable to the wide range of laboratories, studios, or workspaces and have strong online and offline technical support. Lastly, course/application relevance (CAR) is defined as how appropriate the system is with respect to the goals of the course/application. This includes taking into account the HI and SI ratings, ease-of-implementation (EI), and how well they are aligned with the nature of the problem-solving application.

It is important for the designer/instructor to identify clearly the expectation of the course/application. If the goal is to make people think, break problems into systems-level modules and be more independently creative in finding solutions, the popular platforms that have many online available third party sources (reliable and non-reliable ones) would not be necessarily the best choice. However, if the goal is to have a prototyping platform where users solve problems fast and practically, then a platform with a rich set of online/web-based resources provided by third party enthusiasts could be a great choice. Although this could be a part of the evaluation criteria and assessment questions, it is not easily quantified with simple metrics. In addition, the focus and the thematic approach of the resources is usually highly subjective.

This work presents a method for evaluating such systems on the basis of the proposed parameters, so that current and future development systems can be classified and paired with an appropriate problem-solving course or application. Additionally, an example of the method’s use is provided.

II. Experimental Setup

Starter kits of the following embedded system platforms were acquired and experimented with (Figure 1):

- Arduino (http://www.arduino.cc)
- TI Launchpad (http://www.ti.com/launchpad)
- Leaflabs Maple (http://leaflabs.com/devices/maple/)
- .NET Gadgeteer (http://www.netmf.com/gadgeteer/)
- TinkerForge (http://www.tinkerforge.com/)
- Phidgets (http://www.phidgets.com/)

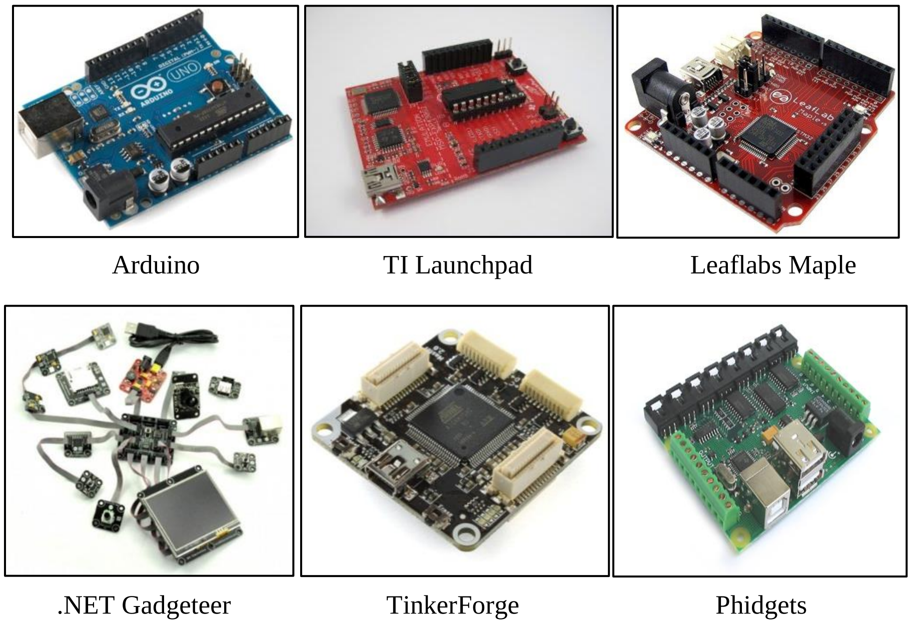


Figure 1: The different embedded systems experimented with.

These devices were evaluated using the questions posed in Appendix A. These questions were chosen specifically to expose the system’s unique attributes. A brief summary of these questions is shown below, with detailed descriptions discussed in the next section:

Hardware-Intensive (HI):

1. Does the system inherently support through-hole (leaded) components?
2. How many components does it take to interface with the controller (wire not included)?
3. Is a separate breadboard needed to interface with discrete components?
4. Is the controller readily programmable via USB?
5. Are the peripheral interface connectors keyed?

Software-Intensive (SI):

1. Is a bootloader required to program the device as the system manufacturer intended?
2. Can the integrated development environment (IDE) provided or suggested by the manufacturer support object-oriented languages?
3. Do the inherent software commands intuitively describe the intended hardware-based result?
4. Can an operating system (like Linux, Android, etc.) be loaded onto the system?
5. Does the IDE provided or suggested by the manufacturer have debugging capability?

Ease of Implementation (EI):

1. How many software packages need to be installed to program the controller?
2. Is the system supported by multiple operating systems?
3. How many commands are required to illuminate an LED?
4. Can the IDE be run from a USB drive?
5. Do the manufacturers provide "Getting Started" (or equivalent) videos on the embedded system website (posts originating from site regulators, not user forums)?

Since the CAR parameter is highly subjective, it is up to the designer/instructor to determine the details of this parameter. This work provides an example of how this can be utilized as described in the "Course/Application Relevance" part of the next section.

As shown above, each parameter has five questions associated with it. For each of question, either 0 points or 1 point is given, depending on the answer (the details concerning specifically how a 0 or 1 is given can be found in Appendix A). The points are then summed, with a total possible rating of 5 per parameter.

III. Discussion of the Measured Parameters

Hardware-Intensive (HI)

Hardware-intensive refers to the level at which a user is required to focus on hardware challenges.

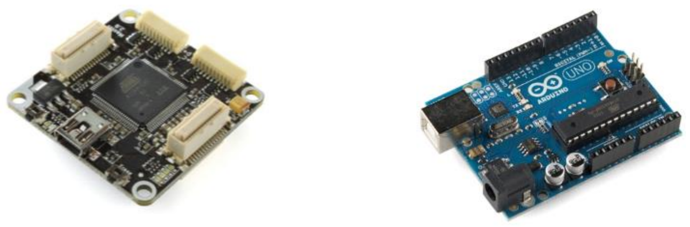


Figure 2: Examples of embedded systems that score high (left) and low (right) for the hardware-intensive parameter.

As an example of the HI parameter, notice the different types of sockets associated with the embedded devices in Figure 2. The device on the left has keyed connectors meant for specialized cables. So, the user does not need to be so aware of how the hardware is connected, and thus it may be fairly straight forward to deal with the hardware associated with his device. This would lead to a low HI rating. In contrast, the device on the right has sockets that are not keyed and their size suggest the availability for discrete component connections (i.e. leaded resistors, capacitors, LEDs, etc.). In this case the user must be cautious about how hardware is connected, which would lead to a higher HI rating.

Further analysis that exposes similar hardware aspects of embedded systems is given in the form of a five-question evaluation. Each question proposed regarding the HI parameter is described below (see Appendix A for further detail).

1. Does the system inherently support through-hole (leaded) components?
Discrete, through-hole components are quite common in hobby electronics and rapid prototyping in engineering education. An embedded system that supports the addition of these components requires more focus on the hardware. In contrast, modules that simply plug into the controller require less focus on hardware.

2. How many components does it take to interface with the controller (wire not included)?
In certain systems, extra modules are required to provide power to the controller as well as additional peripherals. Having a system with on-board power regulation minimizes extraneous connections and the concern of whether or not enough power is provided, diverting focus from the hardware.

3. Is a separate breadboard needed to interface with discrete components?
Some embedded systems have built-in breadboards. This suggests that the supporting documentation will explain how it works. Requiring a separate breadboard assumes the user will know or research how it works, and thus requires more focus on hardware.

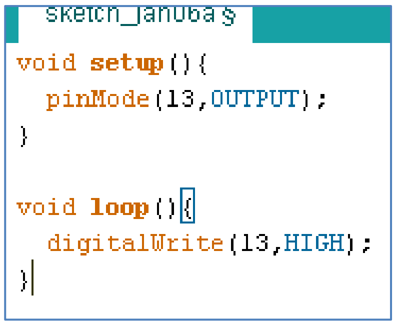
4. Is the controller readily programmable via USB?
Different embedded systems can be programmed in different ways. The most common ways to program a microcontroller from a user's perspective are via in-circuit serial programming (ICSP) and the universal serial bus (USB). Since USB is a widely accepted standard and simply requires a cable (in most cases), having this diverts attention from hardware issues. The ICSP interface requires that the user is familiar the ICSP pins on the controller and has a separate programmer that interfaces with the IDE.

5. Are the peripheral interface connectors keyed?
When interfacing peripherals, having keyed connectors diverts attention from hardware challenges. Connectors that are not keyed require the user to be familiar with power pins, polarity of components, etc.

In this experiment, quantitative conclusions were drawn from the data accumulated from the response to questions in Appendix A. The HI parameter has a rating that can range from 0 to 5, where 0 indicates that the system requires low focus on hardware challenges and 5 requires high focus.

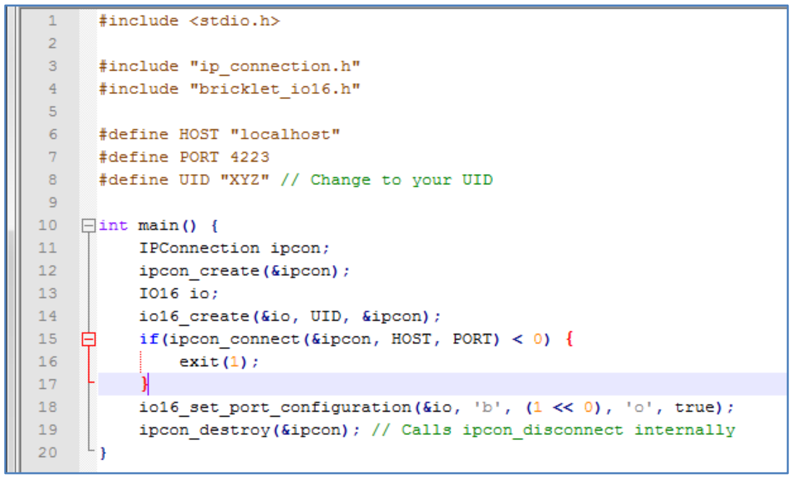
Software-Intensive (SI)

Software-intensive refers to the level at which a user is required to focus on software challenges.



```
void setup(){
  pinMode(13,OUTPUT);
}

void loop(){
  digitalWrite(13,HIGH);
}
```



```
#include <stdio.h>

#include "ip_connection.h"
#include "bricklet_io16.h"

#define HOST "localhost"
#define PORT 4223
#define UID "XYZ" // Change to your UID

int main() {
    IPConnection ipcon;
    ipcon_create(&ipcon);
    IO16 io;
    io16_create(&io, UID, &ipcon);
    if(ipcon_connect(&ipcon, HOST, PORT) < 0) {
        exit(1);
    }
    io16_set_port_configuration(&io, 'b', (1 << 0), 'o', true);
    ipcon_destroy(&ipcon); // Calls ipcon_disconnect internally
}
```

Figure 3: Examples of embedded systems that score high (left) and low (right) for the software-intensive parameter.

As an example of the SI parameter, take note of the two snippets of code in Figure 3. Each snippet corresponds to a different embedded system, both attempting to illuminate an LED with minimum amount of code. The code on the left is short and simple, with commands that intuitively describe the device and its hardware-based actions. The code on the right seems fairly complex for such a minimal task, with commands that are not so intuitive. In this case the embedded system in the left part of Figure 3 would score low in SI, while the system on the right would score high.

Further analysis that exposes similar software aspects of embedded systems is given in the form of a five-question evaluation. Each question proposed regarding the SI parameter is described below (see Appendix A for further detail).

1. Is a bootloader required to program the device as the system manufacturer intended?
Bootloaders are libraries that need to be specially programmed to a controller to allow for a more high-level interface. Burning a bootloader to the device requires an understanding of its use, how to perform such a task, and thus more focus on software.

2. Can the IDE provided or suggested by the manufacturer support object-oriented languages?
Object-oriented languages are intended to relieve the user of certain complications that hardware-level languages face. Thus, a system with object-oriented programming capability allows the user to focus more on software challenges.

3. Do the inherent software commands intuitively describe the intended hardware-based result?
In many embedded systems, the programming languages used are widely known and generally not specific to the device alone. Manufacturers usually provide libraries, header files, etc. that make interfacing to the specific hardware easier. However, it is becoming more common that these well-known languages are being modified or refined to contain commands or structures that are better suited with the hardware. Often times these modifications include commands whose name reflect the desired hardware-based outcome. For example, the command “analogRead(2)” reads the analog voltage at pin 2 on a certain embedded system. This is a built-in function which is pre-installed with the IDE for this device. Systems with built-in commands that intuitively describe the intended result generally allow the user to focus less on software challenges.

4. Can an operating system (like Linux, Android, etc.) be loaded onto the system?
Embedded systems that can have operating systems installed on them generally direct focus on software challenges.

5. Does the IDE provided or suggested by the manufacturer have debugging capability?
Having debugging capability increases potential capabilities and challenges in software.

In this experiment, quantitative conclusions were drawn from the data accumulated from the response to questions in Appendix A. The SI rating ranges from 0 to 5, where 0 indicates that the system requires low focus of software challenges and 5 requires high focus.

Ease-of-Implementation (EI)

An embedded system that is easy to implement has technical support that provides information consistent with observations in lab (i.e. how "trustworthy" the provided information is), requires few steps to install software, and overall has little overhead from the user's perspective. Additionally, for every embedded system experimented with, there are communities that offer technical support or supplementary documentation. However, this support is more intuitive for some embedded systems than for others. These communities can be formed and regulated by the embedded system designers, or it may be formed by a third party. The EI parameter is a measure of the how quickly and easily the embedded system can be used, as well as how versatile the system is in different environments.

Analysis that exposes implementation aspects of embedded systems is given in the form of a five-question evaluation. Each question proposed regarding the EI parameter is described below (see Appendix A for further detail).

1. How many software packages need to be installed to program the controller?
In some cases many different software packages need to be installed with required permissions on the computer. This suggests it is relatively difficult to implement the embedded system, as opposed to those that require only one package.

2. Is the system supported by multiple operating systems?
Having a versatile embedded system that is supported on multiple operating systems can decrease implementation issues for the user.

3. How many commands are required to illuminate an LED?
This question tests how easily the embedded system can access IO ports. In certain systems, this can be more cumbersome than others (Figure 3).

4. Can the IDE be run from a USB drive?
For some systems, the IDE software simply needs to be downloaded and extracted to a folder for it to run and program the device. This capability suggests easy implementation and portability.

5. Do the manufacturers provide "Getting Started" (or equivalent) videos on the embedded system website (posts originating from site regulators, not user forums)?
With recent efforts in the rapid deployment of media, it is often preferred to watch a video on how to get started with an embedded system rather than read a document. A system with reliable and appropriate "Getting Started" videos can be easier to implement than those without.

In this experiment, quantitative conclusions were drawn from the data accumulated from the responses to questions in Appendix A. The EI parameter has a rating that can range from 0 to 5, where 0 indicates that the embedded device is not very easy to implement and 5 means that the device is easy to implement quickly.

It should be noted that the EI parameter is slightly subjective. This work proposes questions that have been influenced by experience in a standardized lab environment. In many of these environments, there is little control of the workstation file system permissions. Therefore, in addition to how easy the embedded system is to use from a user's perspective, the perspective of a lab instructor is also taken into consideration.

Course/Application Relevance (CAR)

The CAR parameter refers to whether or not the embedded system is suited for the course being offered or the application at hand. Therefore, this parameter is subjective based on the purpose of the course or application. For example, a class that aims to focus primarily on hands-on circuit-oriented problem solving may favor an embedded system that scores a high HI parameter. Alternatively, a class that stresses programming technique and syntax may favor a system scoring a high SI parameter. Overall, the CAR parameter is a personal judgment based on the embedded system's practicality with respect to objectives of the course or application offered.

In this experiment, concluding results were based on a course offered for freshman electrical engineers at Iowa State University. In this course, one of the main objectives is to emphasize creative systems-level problem solving using critical thinking. Additionally, it is desired to have hands-on experience with circuits and measurement devices. The authors decided that the appropriate embedded system for this course must have an EI rating no less than 3, an HI rating greater than or equal to 3, and an HI rating greater than or equal to the SI rating.

IV. Results

Based on the data accumulated from the responses to questions proposed in Appendix A, conclusions were drawn about the HI, SI, and EI parameters (Table 1). These parameters played a role in deciding whether or not the system was relevant to a freshman electrical engineering course at Iowa State University.

Embedded System	HI	SI	EI	CAR (Y/N)
Arduino	🟩🟩🟩⬜⬜	🟩🟩⬜⬜⬜	🟩🟩🟩🟩⬜	Y
TI Launchpad	🟩🟩🟩⬜⬜	🟩🟩🟩⬜⬜	🟩🟩🟩🟩⬜	Y
Leaflabs Maple	🟩🟩🟩⬜⬜	🟩🟩⬜⬜⬜	🟩🟩🟩🟩⬜	Y
.NET Gadgeteer	🟩🟩⬜⬜⬜	🟩🟩🟩🟩⬜	🟩⬜⬜⬜⬜	N
Tinkerforge	🟩🟩⬜⬜⬜	🟩🟩⬜⬜⬜	🟩⬜⬜⬜⬜	N
Phidgets	🟩⬜⬜⬜⬜	🟩🟩🟩🟩⬜	🟩🟩⬜⬜⬜	N

Table 1: Quantified data based on results of Appendix A questions

It was found that of the embedded systems experimented with, the Arduino, TI Launchpad, and Leaflabs Maple were suited for the course described. Although the .NET Gadgeteer, Tinkerforge, and Phidgets systems are versatile and highly capable, they did not score well in this study for EI, nor did they meet the overall criteria put forth by the authors. For systems that score low on both HI and SI, it does not necessarily mean that they are neither hardware- nor software-intensive systems. It only means that with respect to a learner-centric environment, the systems may not be well suited. There are many other systems available as well that were not included in this study, so the presented data serves as preliminary results.

V. Conclusion

In this work, a novel and practical method for categorizing embedded systems was proposed, and implementation and assessment were performed in a freshman Electrical Engineering classroom at Iowa State University. Results from applying this method showed that the Arduino, TI Launchpad, and Leaflabs Maple embedded systems were well suited for a course that aims to emphasize creative systems-level problem solving using critical thinking and hands-on, hardware-based activities. Competitive products, such as the .NET Gadgeteer, Tinkerforge, and Phidgets systems were found to be not as well suited based on this evaluation method. There are many other competitive systems that were not evaluated, so the presented data serves as preliminary results.

With the newly proposed method, and given the purpose and goals of a course or application, objective research about an embedded system could predict, in general, overall level of effort. This could allow for further confidence in choosing an embedded system without the time-intensive task of experimenting with the many systems available.

VI. Acknowledgements

The authors would like to acknowledge Dr. Phillip Jones for excellent insight and fruitful discussions and also students Riley O’Connor, Yan Yao Chen, Jui Yen Chua, Haisong Lin, Jeremy Vens, and Matt Dresser for their kind patience and help with this study.

This work was supported by the National Science Foundation under awards: DUE 0837314 and DUE-0920164. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

VII. References

[1] Hall, T.S.; Hamblen, J.O., "System-on-a-programmable-chip development platforms in the classroom," Education, IEEE Transactions on , vol.47, no.4, pp.502,507, Nov. 2004

[2] Al-Busaidi, A.M., "Development of an educational environment for online control of a biped robot using MATLAB and Arduino," Mechatronics (MECATRONICS) , 2012 9th France-Japan & 7th Europe-Asia Congress on and Research and Education in Mechatronics (REM), 2012 13th Int'l Workshop on , vol., no., pp.337,344, 21-23 Nov. 2012

[3] Neto, J. M.; Paladini, S.; Pereira, C.E.; Marcelino, R., "Remote educational experiment applied to electrical engineering," Remote Engineering and Virtual Instrumentation (REV), 2012 9th International Conference on , vol., no., pp.1,5, 4-6 July 2012

[4] Ogawa, H.; Oguntoyinbo, B.; Tochi, K.; Naoe, N., "Electric vehicle project for introduction to engineering Creation Experiment III," Engineering Education (ICEED), 2011 3rd International Congress on , vol., no., pp.28,31, 7-8 Dec. 2011

[5] Hodges, S.; Taylor, S.; Villar, N.; Scott, J.; Bial, D.; Fischer, P.T., "Prototyping Connected Devices for the Internet of Things," Computer , vol.46, no.2, pp.26,34, Feb. 2013

[6] Hodges, Steve; Villar, N.; Scott, J.; Schmidt, A., "A New Era for Ubicomp Development," Pervasive Computing, IEEE , vol.11, no.1, pp.5,9, January-March 2012

[7] Balid, W.; Alrouh, I.; Hussian, A.; Abdulwahed, M., "Systems engineering design of engineering education: A case of an embedded systems course," Teaching, Assessment and Learning for Engineering (TALE), 2012 IEEE International Conference on , vol., no., pp.W1D-7,W1D-12, 20-23 Aug. 2012

[8] Takayama, Y.; Koga, T.; Nitta, T.; Yanagisawa, H.; Shigemura, T., "Curriculum design for engineering education on embedded system based on broad partnership with university, corporation and local school," Teaching, Assessment and Learning for Engineering (TALE), 2012 IEEE International Conference on , vol., no., pp.T1D-1,T1D-7, 20-23 Aug. 2012

[9] Li Xiaojuan; Guan Yong; Yuan Huimei, "Curriculum Development and Progressive Engineering Practice Design in Embed System Education," Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on , vol., no., pp.228,232, 12-15 Oct. 2008

[10] Li Xiaojuan; Guan Yong; Yuan Huimei, "A Novel Course System and Engineering Practice Design for Embed System Education," Computer Science and Software Engineering, 2008 International Conference on , vol.5, no., pp.1388,1391, 12-14 Dec. 2008

[11] Kodama, T.; Suzuki, Y.; Chiba, S., "Development of a remote practice system for embedded system education," Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on , vol., no., pp.53,58, 15-17 July 2010

[12] Pimentel, A.D.; Erbas, C.; Polstra, S., "A systematic approach to exploring embedded system architectures at multiple abstraction levels," Computers, IEEE Transactions on , vol.55, no.2, pp.99,112, Feb. 2006

[13] Djordjalian, A.; Lutenberg, A.; Cruz, J.M.; Garcia, S.; Martos, P.; Gomez, P., "Developing an intermediate embedded-systems course with an emphasis on collaboration," Frontiers in Education Conference (FIE), 2011 , vol., no., pp.F3H-1,F3H-8, 12-15 Oct. 2011

[14] Xiumin Shi; Ji Zhang; Yanbing Ju, "Research and Practice in Undergraduate Embedded System Course," Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for , vol., no., pp.2659,2663, 18-21 Nov. 2008

Appendix A: Questions for and Results of Quantitative Analysis

Hardware-Intensive (HI)
The questions that led to the quantitative conclusions for hardware-intensive are the following:

1. Does the system inherently support through-hole (leaded) components?
    a. Yes (1 points)
    b. No (0 point)
2. How many components does it take to interface with the controller (wire not included)?
    a. 1 (0 points)
    b. Greater than 1 (1 point)
3. Is a separate breadboard needed to interface with discrete components?
    a. Yes (1 points)
    b. No (0 point)
4. Is the controller readily programmable via USB?
    a. Yes (0 points)
    b. No (1 point)
5. Are the peripheral interface connectors keyed?
    a. Yes (0 points)
    b. No (1 point)

Software-Intensive (SI)
The questions that led to the quantitative conclusions for software-intensive are the following:

1. Is a bootloader required to program the device as the system manufacturer intended?
    a. Yes (1 point)
    b. No (0 points)
2. Can the IDE provided or suggested by the manufacturer support object-oriented languages?
    a. Yes (1 point)
    b. No (0 points)
3. Do the inherent software commands intuitively describe the intended hardware-based result?
    a. Yes (0 point)
    b. No (1 points)
4. Can an operating system (like Linux, Android, etc.) be loaded onto the system?
    a. Yes (1 point)
    b. No (0 points)
5. Does the IDE provided or suggested by the manufacturer have debugging capability?
    a. Yes (1 point)
    b. No (0 points)

Ease of Implementation (EI)
1. How many software packages need to be installed to program the controller?
    a. Only 1 (1 point)
    b. Greater than 1 (0 points)
2. Is the system supported by multiple operating systems?
    a. Yes (1 point)
    b. No (0 points)
3. How many commands are required to illuminate an LED?
    a. Less than or equal to 6 (1 point)

    b. Greater than 6 (0 points)
4. Can the IDE be run from a USB drive?
    a. Yes (1 point)
    b. No (0 points)
5. Do the manufacturers provide "Getting Started" (or equivalent) videos on the embedded system website (posts originating from site regulators, not user forums)?
    a. Yes (1 point)
    b. No (0 points)

Results

Arduino

Question	**HI**	**SI**	**EI**
1	1	1	1
2	0	1	1
3	1	0	1
4	0	0	1
5	1	0	0
Total	3/5	2/5	4/5

.NET Gadgeteer

Question	**HI**	**SI**	**EI**
1	0	1	0
2	1	1	0
3	1	1	0
4	0	0	0
5	0	1	1
Total	2/5	4/5	1/5

TI Launchpad

Question	**HI**	**SI**	**EI**
1	1	0	1
2	0	1	1
3	1	1	1
4	0	0	0
5	1	1	1
Total	3/5	3/5	4/5

Leaflabs Maple

Question	**HI**	**SI**	**EI**
1	1	1	1
2	0	1	1
3	1	0	1
4	0	0	1
5	1	0	0
Total	3/5	2/5	4/5

Tinkerforge

Question	**HI**	**SI**	**EI**
1	0	0	0
2	1	1	1
3	1	0	0
4	0	0	0
5	0	1	0
Total	2/5	2/5	1/5

Phidgets

Question	**HI**	**SI**	**EI**
1	0	1	0
2	0	1	1
3	1	0	1
4	0	1	0
5	0	1	0
Total	1/5	4/5	2/5