Applied Laser Technology

Vol. 30, No.2, APRIL (2023), pp.01–06

Helping Students and Professionals Simplify Electronics Design with the SHF: SMALL Framework

M VENKATA RATNAM[1], P ANIL KUMAR REDDY[2]

Assistant professor [12]

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

P.B.R. VISVODAYA INSTITUTE OF TECHNOLOGY & SCIENCE S.P.S. R NELLORE DIST, A.P, INDIA, KAVALI-524201

Introduction

There has never been a better moment for people to express their own creativity. Making one's own hardware, software, clothing, and even art and furnishings has become hip in recent years thanks to the "Maker" movement. Although many people like the creative process of manufacturing, the entrance barrier is sometimes high for complex projects, especially those involving electronics. Simple electronics are often all that is required to finish a job. One approach to show that electronics requirements are often rather modest is to peruse projects on sites like Hockaday [?] and Kickstarter [?]. The development of such items usually entails little more than the mechanical application of well-established technological ideas. Most projects don't need much in the way of electronics expertise, and the ones that do have quite simple electronics requirements. Creating such gadgets is challenging even when based on proven technological principles.

The designer faces an enormous learning curve in order to figure out which principles are applicable and how to apply them. This includes, but is not limited to, learning how to design electronic circuits, how to construct printed circuit boards (PCBs), becoming acquainted with readily available electronic components, and how to write computer code for micro controllers and/or other computing platforms. We propose that this repetitious technical work be automated in whole or in part in order to free up the designer's time for more creative pursuits. We will standardise engineering concepts in a software tool chain to make it much easier to develop elementary electrical systems. Expert designers may use the same set of tools to swiftly build complex systems from a wide variety of configurable, adaptable components that share clear APIs. Our technology will help designers of all levels save time while developing electrical circuits for a wide range of applications. We want to do this by raising the bar at which design requirements are specified and system solutions are implemented. The final result will be a cutting-edge set of resources for designing circuits, specifying their requirements, and recycling its constituent components. Hardware Meant EZ has two separate user interfaces that are made for both novice and expert users.

Background

The electronics of a device might be easy to design or quite difficult. There's a way in which this is a problem: In order to create a functional printed circuit board, one must be well-versed in electrical circuits, microcontroller programming, and the available electric components, as well as PCBs, PCB design tools, and best practises in generating PCB schematics and board layouts (resistors, capacitors, integrated circuits, etc.). Since modern PCB design tools give limited help for design re-use, a beginner designer must battle with every aspect of the design, from what high-level activities the circuit must accomplish to the part number for each particular electrical component. One may also say that designing circuits is a piece of cake: Using common design principles and circuit design idioms, an experienced engineer may quickly and easily construct a working device. While building a new circuit, the designer often referred to this "mental library" of past designs (for example, power supply or microcontroller settings).

Hardware Made EZ sets out to address the issue of how a less-skilled designer might take use of an expert's knowledge, experience, and "mental library" to build complicated circuits with high reliability and speed without having to micromanage every detail of the design. In a similar vein, veteran designers may focus on intricacies of technological problems that need fresh perspectives and creative thinking. Hardware Made EZ

ISSN: 1000-372X ALT

Copyright ©2023

divides the traditional device development process into three manageable steps: device specification, architecture design, and build.

Hardware Simplicity

Thankfully, Hardware Made EZ will address the issues and capitalise on the benefits outlined above. For success, we'll be using the following principles of design:

Ability levels vary Hardware Made EZ contains tools that are useful for designers of all skill levels. While novices may quickly develop simple devices, experts may quickly design and execute complex ideas.

It is possible for adoption to happen in phases. Users will be able to embrace our tools and concepts in reasonable pieces without having to repeat their earlier design work since Hardware Made EZ is compatible with the current approach of building devices.

The Multi-Modal Design Hardware Made EZ programme may be used by designers at any point in the process (i.e., between specification and architecture design and between architect true design and implementation).

Language-specific development environments Hardware Made EZ's expressive design language will make it easy to build functioning devices by emulating common practises in software engineering and programming languages, such as interfaces, types, and aspects.

Methods from modern software engineering, like modularity, information hiding, and class-based design, will be included into Hardware Made EZ to facilitate hardware reusability and testing.

How to Create Your First Device

The technological difficulties involved in constructing even a very basic gadget might be daunting for inexperienced builders. Jet is a simple "drag-and-drop" tool for creating and programming devices that will be developed to meet the demands of this demographic by allowing designers to concentrate on the design of the device rather than the implementation concerns. A mock-up of the proposed instrument is shown in Figure 2. A 3D model of the gadget is rendered, and the user is given the option of incorporating their own aesthetic preferences into the final product. Elements of design encompass anything from the device's shape and colour to how it communicates with other devices and the environment. Elements like as buttons, screens, casing cut-outs, and wires are included here (e.g., USB). Unseen components are also a part of the design. For a clock gadget, for instance, the user may demand that it be able to accurately display the time. Designing in a timekeeping feature is important even if it doesn't change how the product looks.

C-Code

The current state of the art in characterising the electrical connections between PCB components is mostly dependent on a technique called "schematic capture." Lines placed between schematic symbols denote the connections between certain components. Because the schematic designer may depend on a wide variety of visual idioms to convey the intended meaning of the schematic to the reader, schematic capture is an easy way to describe even the most basic circuits. Figure 3 depicts the plan for a beginner's remote-controlled "quadcopter" created in a senior-year design course. The electronics of a quadcopter consists of multiple mainly autonomous parts, including a microcontroller, an accelerometer and gyroscope, a power source, four motor controllers, and a number of interfaces for programming and debugging. The designer has used dashed lines to visually distinguish these components and has included text in the schematic to offer further information about each one. Creating, modifying, and maintaining this schematic was more troublesome than it needed to be for a number of reasons. To begin, each part is versatile enough to be employed in the construction of several additional tools. This schematic really borrows elements from publicly available hardware projects. Since any flaws in the source were also copied, this design will not benefit from any updates or corrections made to the source. Second, four identical motor drivers are included into the design. This component of the design ended up needing many tweaks throughout development, and updating all four at once was a time-consuming, error-prone procedure. Third, there are three LEDs and current-limiting resistors included into the design. Due to their unique colour temperatures, each LED has a variable voltage need, thus the resistor values must be adjusted accordingly. In the course of design, we switched from a 3.3 V supply to a 2.7 V supply, necessitating a

ISSN: 1000-372X ALT

Copyright ©2023

recalculation of the values of the corresponding resistors. The motor's current draw, its inductive qualities, and the battery type are all factors that affect the motor control system in a similar way.

This Is the C-Standard Code Library

C-primary Code's objective is design reuse. C-Code will offer a standardised library of parameterizable, re-usable components to showcase the power of design reuse in device design, serve as a resource for users seeking inspiration as they create their own reusable components, and ease the learning curve for newcomers. Power sources, microcontrollers, sensors, motor controllers, wireless communication modules, debugging interfaces, and user interface components will all be part of this (e.g., buttons, knobs, displays, and LEDs). Using the LED from Figure 3 as an example, this section of the library demonstrates how to utilise a component. The bare minimum for controlling an LED is a power supply that outputs V dd and a connection to ground. The LED is accompanied by a current-limiting resistor, R1, which is also included. Each LED has its own maximum forward current (Imax) and forward voltage (Fwd.). Engineers may utilise Ohm's law to settle on a value for R1's resistance.

$$R1 = \frac{V_{dd} - V_{LED}}{I_{max}}$$

Learning with Jet and Beyond

To prove that Jet has been successful in its mission to make device design simpler for non-experts, we need to get it into the hands of those non-experts and utilise their comments to shape future iterations of the software. One way we want to achieve this goal is by incorporating Jet into a course offered at UC San Diego, as well as by making Jet publicly available and carefully considering the responses it receives.

Robotic Marching Band during Campus Rally

One of our beginning computer science courses will include a companion course named "Robot Parade" where Jet will be used (known as CSE11 at UCSD). The CSE11 course covers the fundamentals of programming and objects over the course of a quarter (ten weeks). The engineering school at UCDSD is embarking on a massive programme to introduce project-based learning to first-year students, and Robot Parade will play a key role in this effort. During the duration of this grant, we want to provide Robot Parade at least once every year and to expand the course so that it is available to a large percentage of freshmen. Unlike CSE11, Robot Parade offers pupils a number of benefits: To facilitate more student-instructor engagement, Robot Parade will be delivered in smaller groups.

As a result of participating in Robot Parade, students will be able to put their newly acquired programming abilities to use in an exciting new setting (i.e., programme Ming’s robots).

Students will learn how to collaborate on a programming project in a cooperative setting using Robot Parade.

Robot Parade will provide children the opportunity to learn how to code by creating robots that can interact with their surroundings.

• Students participating in Robot Parade will get exposure to the intersection of coding and the creation of tangible items. • Students participating in Robot Parade will be required to consider design restrictions from several perspectives.

These objectives will be met via the use of hands-on programming projects, brief lectures, and actual design work throughout the course of Robot Parade. Following is a course overview and explanation of how we want to assess Robot Parade's usefulness.

General Description of the Contents Covered in Class

In a structured learning lab setting, students will spend 10 weeks building and programming basic robots based on the Arduino platform. The final project for the class is a collaborative effort in which students work in small groups to build and develop a robot to carry out an assigned job.

ISSN: 1000-372X ALT

Copyright ©2023

Course Objectives and Overview in Week 1 Group contemplation about the nature and purpose of robots.

• Weeks 2-4 Students will learn how to programme a robot's many "components" each week. Motors, lights, sensors (bumper, distance, servo, speaker) are all examples of parts. Robotics programming techniques will be discussed weekly. Minimal feedback control systems and finite state machines are two such examples.

• In the fifth week, students utilise Jet to create a robot that makes use of the knowledge they've gained in previous weeks. The design will be limited by the Arduino microcontrollers that act as the robot's "brain," as well as by size and cost considerations.

In Week 6, students will learn how to solder and assemble their robots.

• Week 7-10 Each group of students will be responsible for programming their own robot to carry out an individual assignment. The final event of the course is called "Robot Parade," and it consists of students presenting their finished robots to the rest of the group. There will be no restrictions on who may attend the presentation, so feel free to invite your colleagues.

Analysis of the Robot Parade

The purpose of Robot Parade is to increase learning and retention rates among a wide variety of incoming freshman. Two surveys, the Computing Attitudes Survey [?] and the Computer Science Attitudes Survey [?], will be given to students who take the course in order to gauge its success. Students enrolled in CSE11 but not Robot Parade will also be given the questionnaires to serve as a control group. The Internal Review Board at UC San Diego will oversee our work in creating the experimental procedure. We will modify future iterations of Robot Parade based on the results of these polls, our classroom experience, and feedback from students.

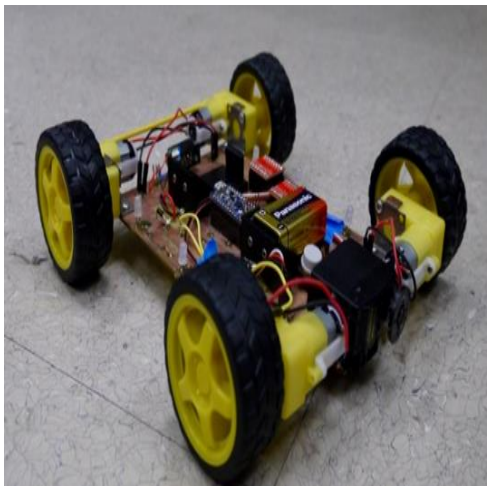

Figure 4: An Experimental Robotic Marching Unit One of the robots created by first-year students for the Robot Parade pilot. The student said his or her interest in creating and constructing robots was a major motivating element in continuing with the CS major.

Success in the Early Stages

The effectiveness of memory-enhancing courses like Robot Parade has previously been shown. Our group's lab took part in the department's Summer Program for New Students this past summer (SPIS). In order to "jump start" their education, SPIS brings accepted first-year students to campus for a 4-week course in computer science. Four SPIS students were given the opportunity to participate in a project similar to the Robot Parade. Under my supervision, my graduate and undergraduate research students construct robots utilising more traditional design tools than Jet. The robot in Figure?? is an example of what they created. Enjoyable times were had by all of the pupils. One of the SPIS group members was having a lot of trouble in CSE11 and was thinking about dropping out of school last week, I found out. But she said that working with robots at SPIS was a big reason why she decided to stick with the field. We believe that Jet will allow us to provide many more students

ISSN: 1000-372X ALT

Copyright ©2023

with design-based, hands-on experiences, with the added bonus of maybe convincing some of them to continue studying computer science.

Alternative Campus Activities

The COSMOS programme is the California State Summer School for Mathematics and Science, which is held at UCSD annually. Comparable to SPIS, COSMOS welcomes a diverse student body. For COSMOS next summer, the PI has already proposed a Jet-based module. An on-campus "Maker Space" is now being constructed at UC San Diego for pedagogical purposes and student-initiated research and design initiatives. The PI is on the Maker Space's steering committee and hopes to collaborate with other educators to identify promising uses for Jet and the other products we develop. Tools will develop in response to this data.

Community Engagement Through Elementary, Middle, and High Schools

We want to utilise and improve Jet in tandem with local schools and other organisations concerned with education (such as local "Maker spaces"). Educators from many nearby magnet schools have been contacted about maybe incorporating Jet into their lessons or after-school clubs as a result of our pitch for Jet and Robot Parade. As soon as Jet is ready, we'll collaborate with these teachers to create Jet-enabled lessons that are suited to their students' needs and interests, and then help them distribute those lessons to their students. We also want to publish Jet online for free usage by anybody, wherever in the world, for the purpose of device creation. Users will likely put Jet to use in ways we never imagined, and their input on how to make Jet better will be vital to us.

The Outcomes of NSF Past and Present Support

For his work on "SHF: Small: Reengineering Database Systems for Fast SSDs," Steven Swanson has received funding from the National Science Foundation (NSF) in the form of grant CCF-1219125 ($500,000.00, 7/1/2012-6/30/2015). The goal of this project is to rebuild existing database systems so that they may make use of new forms of non-volatile memory.

Brains above brawn

This study took use of new forms of sophisticated non-volatile memory to speed up data management operations with a heavy throughput. A unique database logging protocol and a novel design for a programmable SSD that can speed up database workloads are the results of this study. Three high-quality papers [3, 4, 5] were a direct outcome of our study. The papers may be found on the PI's website for free download.

deeper repercussions

Students have been given doctorate and master's degrees for work that is directly related to this investigation. All four students' efforts will ultimately benefit their individual dissertations.

excellence in ideas and general applicability

Mind over muscle

The proposed study adapts tried and true procedures from the disciplines of software engineering and system design to the design of electrical systems. Several of the PI's large-scale research initiatives were successful because they used consistent methodologies across both hardware and software. The PI has also been a speaker and directed the creation of some complex PCB designs. For the last two years, he has worked in a classroom/lab setting to instruct master's and undergraduate students in the fundamentals of electrical device construction. Expertise and access to powerful computer capabilities in all of these areas are available via the PI's research group in Computer Science and Engineering and the San Diego Supercomputing Center.

consequences for a broader audience

The proposed work will have far-reaching impacts in two ways: it will make it easier for designers of all stripes to build electronic systems effectively, and it will increase access to hands-on, project-based training experiences in hardware and software design. When finished, the intended work will result in tools that are expected to have a major impact on PCB design as a field. Our goal is to make the resources widely available,

ISSN: 1000-372X ALT

Copyright ©2023

so they may be used by as many people as possible, and so they will be interoperable with other open-source or freely available software. If we include these technologies into our lesson plans, students from all walks of life will have a firm grasp on both hardware and software. Early results from our experiments with hardware creation among, say, college freshmen, suggest that it may be an efficient instrument for bolstering self-assurance and assuring long-term success in the field**.**

References

[1] Adafruit Industries. http://www.adafruit.com.

[2] Spark Fun Electronics. https://www.sparkfun.com.

[3] J. Coburn, T. Bunker, M. Shwarz, R. K. Gupta, and S. Swanson, "From ARIES to MARS:transaction support for next-generation solid-state drives," in Proceedings of the 24th International Symposium on Operating Systems Principles (SOSP), 2013.

[4] S. Seshadri, M. Gahagan, S. Bhaskaran, T. Bunker, A. De, Y. Jin, Y. Liu, and S. Swanson, "Willow: A user-programmable ssd," in Proceedings of the 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI '14), 2014.

[5] R. Balasubramonian, J. Chang, T. Manning, J. H. Moreno, R. Murphy, R. Nair, and S. Swanson, "Neardata processing: Insights from a micro-46 workshop," Micro, IEEE, vol. 34, pp. 36–42, Ju

ISSN: 1000-372X ALT

ISSN: 1000-372X ALT

Copyright ⓒ2023