



A Hands-on Introduction to Embedded Systems & IOT

Dr. James Peyton Jones, Villanova University

James Peyton Jones is a professor of electrical and computer engineering, and a member of the Center for Nonlinear Dynamics and Control at Villanova University.

A Hands-on Introduction to Embedded Systems & IOT

J.C. Peyton Jones

Department of Electrical & Computer Engineering, Villanova University, Villanova PA 19085

Abstract

Traditional curricula often teach low-level fundamentals of digital logic before introducing students to the ‘higher-level’ topics of microprocessors and the Internet Of Things (IOT). An alternative and potentially more motivating approach is to reverse this sequence. This paper describes the design of a new hardware kit and sequence of laboratory exercises which aim to give students hands-on experience with Embedded systems and IOT at an early stage in their academic careers. The kit is based on a low-cost, wireless-networked, 32-bit ARM microcontroller with integrated Cloud support. The sequence of lab exercises which build incrementally on one another is described in detail, and the experience gained running them for the first time is reported. Outcomes relate to the ability to extend knowledge from an associated lecture course in order to apply, analyze and create embedded / IOT solutions to practical engineering problems. Rubric-based assessment demonstrates that the large majority (>90%) of the class met these outcomes. A further outcome relates to improved motivation and learning through reversing the traditional course sequence. Assessment of this outcome is still a work in progress but initial survey results are encouraging and show that the course was well received.

Introduction

Embedded systems and the Internet of Things (IOT) are becoming increasingly important with approximately the same number of IOT devices as people on the planet [1], so it is important to reflect these changes in the Engineering Curriculum. Traditional courses tend to adopt a ‘bottom-up’ approach, starting with elementary logic gates and Boolean algebra and only later in the curriculum reaching the level of microprocessor development or networked applications [2], [3]. The recent availability of increasingly powerful, yet low-cost, wireless networked devices makes it much more feasible to adopt a ‘middle-out’ approach, where students get to experience and use embedded systems and IOT, before delving deeper into the details of how such systems are constructed at the circuit level or integrated in a networked environment. This paper describes a new hardware kit and sequence of exercises which aim to support this approach by giving students hands-on experience with Embedded systems and IOT at an early stage in their academic careers. Based on the rapid growth and enthusiasm observed in the ‘maker’ community, it is postulated that this approach will help engage students and motivate them for later, more in-depth courses in the field. Alternatively, for those students who have only limited opportunities for further study in this discipline (such as Mechanical Engineering students or Mechatronics minors) it is postulated that the new course is more practical and relevant to their needs compared to a traditional first course in fundamental Digital Logic Design. Finally, it should be noted that the kit is fully mobile so for most (though not all) of the lab assignments, students are not constrained to working in the traditional laboratory environment. This aspect proved critical in the response to the COVID-19 crisis since students were able to continue the lab assignments in their home environments.

The need to introduce IOT into the curriculum has been recognized by a number of authors: A freshman year initiative is described in reference [4] where four introductory IOT labs and a

project are introduced in a C++ programming course. The labs were shown to increase student motivation, and ongoing work was directed at developing lighter versions of the labs for non-ECE students, or high school students. At the other end of the spectrum, a higher-level IOT course is described in reference [5] targeting juniors and seniors in CS, CE, and EE. The course used a single custom hardware device (a smart desk-lamp) and focused more on the high-level aspects of IOT, e.g. MQTT brokers, web service providers and frameworks, data analytics and security. Another junior/senior level course is described in reference [6] where a more traditional computer networking course is revamped to gear all the topics towards IOT. The current work is perhaps most similar to reference [7], though this was designed for 2-year Technical Community College students and used a hybrid hardware platform made up of an Arduino and Raspberry Pi.

In this work, the aim was to develop a sophomore level course targeted primarily at 4-year BS EE students, but also taken by some ME students as part of a Mechatronics minor. The course therefore addresses a potential gap between some of the prior works discussed above. Most students had already taken a C programming course in their freshman year, and this meant that there was more time to address fundamental principles and hardware interfacing than might be possible in a freshman course, but it was also not appropriate to go into too much detail on networking, data analytics or security.

Specifically, the course aims to introduce some of the fundamentals of computing (basic computer architecture, memory, number systems, elementary Boolean logic etc.) while also teaching the practicalities of interfacing, interrupt handling, cloud programming, serial communications (UART, SPI, I2C etc.), and the programming of finite state machines. Initially students explore the input/output (I/O) capabilities of the device, focusing on digital, analog, and PWM signals, and using these to control LED's, buzzers, and small dc motors. This work culminates in a project where the students connect their boards to a small mobile robot and encode a simple line-following application. Students then explore interrupt handling and communications, first with cloud-based systems using their boards to send notifications via IFTTT to their gmail or mobile phone accounts, and then more locally using serial protocols to communicate between devices and smart sensors. Finally, the labs conclude with an open-ended burglar alarm system project, based heavily on finite state machines. The base project uses a reflective light sensor to detect intruders and sounds an alarm when armed, but more sophisticated implementations require cloud-based notification and logging, with remote or local security-key entry required in order to control the system. This paper provides details of the lab design, and the experience gained running it for the first time in Spring 2019. Assessment by means of post-course survey and lab completion data shows that the course was well received with very positive results.

Choice of microcontroller

A critical choice in the course design was the microcontroller platform to be used. A large variety of devices are available, ranging from very low cost 8-bit machines to more modern 16- and 32-bit systems. Examples include the Board Of Education (BOE) robot, based on an 8-bit Basic Stamp microprocessor, [8], [9]. Arduino boards, based on 8-bit Atmel microcontrollers have also become popular in the academic community [7],[10], while other educators have based their work on 8- or 16-bit MicroChip PIC devices [11],[12].

A more recent trend, however, has been a shift towards 32-bit ARM-based devices, including for example ARM-based Arduino designs, [13]. There are several advantages to this approach. First, unlike many of the other devices mentioned above, the ARM architecture is not limited to a single semiconductor manufacturer but is a licensed technology which is used by a broad range of suppliers. Its RISC architecture is very elegant, making it a common choice for courses on Computer Architecture, as well as a dominant force in industry. Indeed, it is claimed that over 10 billion ARM devices are sold every year, [3].

As indicated above, however, there is still a very large array of 32-bit ARM based boards to choose from. Table 1 lists a small selection of these. All have similar cost, but differ somewhat in terms of clock speed, RAM, Flash, Communication and I/O capability. The BBC microBit, used particularly in the UK, [14], is the lowest cost but least capable of these devices. The STM32F4 Discovery board, [15],[16], is considerably more powerful, with floating point capability and extensive I/O, but it lacks the Wi-Fi capability that allows seamless transition into the world of IOT. Of the two remaining devices, the Raspberry Pi ZeroW is evidently the more powerful computing device but it is more of a microcomputer – running the Linux operating system, utilizing an SD-card-based file system and even incorporating a camera interface. The Photon, by contrast, is a powerful microcontroller device whose GPIO capability includes Analog to Digital and Digital to Analog converters in addition to multiple PWM, SPI, I2C, I2S, UART and CAN interfaces. The device is specifically designed for IOT applications with ‘Over The Air’ (OTA) programming and cloud support functionality, while still providing access to low-level hardware.



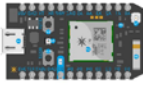

Micro:bit	STM32F4 Discovery	Photon	Rpi-Zero W
			
\$15	\$20	\$19	\$10+\$9
ARM-M0	ARM-M4 + FPU	ARM-M3	ARM11 + VFP
16 MHz	168 MHz	120 MHz	1 GHz
16 KB RAM	192 KB RAM	128 KB RAM	512 MB RAM
256 KB Flash	1 MB Flash	1 MB Flash	SD Card (16GB)
USB, Bluetooth	USB OTG	USB, WiFi	USB OTG, WiFi, Bluetooth, HDMI
19xGPIO = I2C, SPI, 6xADC + 3D Accel, 3D Mag 25xLED array 2xBtn	82xGPIO= 3xI2C, 3xSPI, 2xI2S, 16xADC, 2xDAC, 2xCAN, 4xUART, Ethernet + Microphone + Audio DAC 3D Accel, 8xLED, 2xBtn	18xGPIO = I2C, I2S, 2xSPI, UART, CAN 8xADC, 2xDAC 9xTimer/PWM	26xGPIO = 2xI2C, 2xSPI, UART 2xPWM Camera

Table 1: A comparison of several low-cost ARM-based boards

The OTA and cloud support functionality is illustrated in Fig. 1. Users develop source code in either a cloud-based, or locally installed Integrated Development Environment (IDE). [The low-level Device OS uses the same Wiring code framework as Arduino]. By default, all programs are transmitted to the Particle Cloud for compilation, and the resulting executable binaries then downloaded wirelessly OTA to the Photon. No physical connection between the development platform and the Photon target is needed although user programs can communicate with a host over USB if desired. User programs can also communicate dynamically with the Particle Cloud, and hence to a large number of web services as indicated in the top-right corner of Fig. 1. Finally, developers can monitor this traffic, publish events or make PUT / GET http requests by logging into a Particle Cloud Console.

This combination of hardware capability and cloud integration resulted in the Photon being chosen as the primary platform for this course. Its small physical footprint, 5V tolerant pins and stable design were also significant considerations since robustness and the ability to fit all needed components on a half-size breadboard were considered critical.

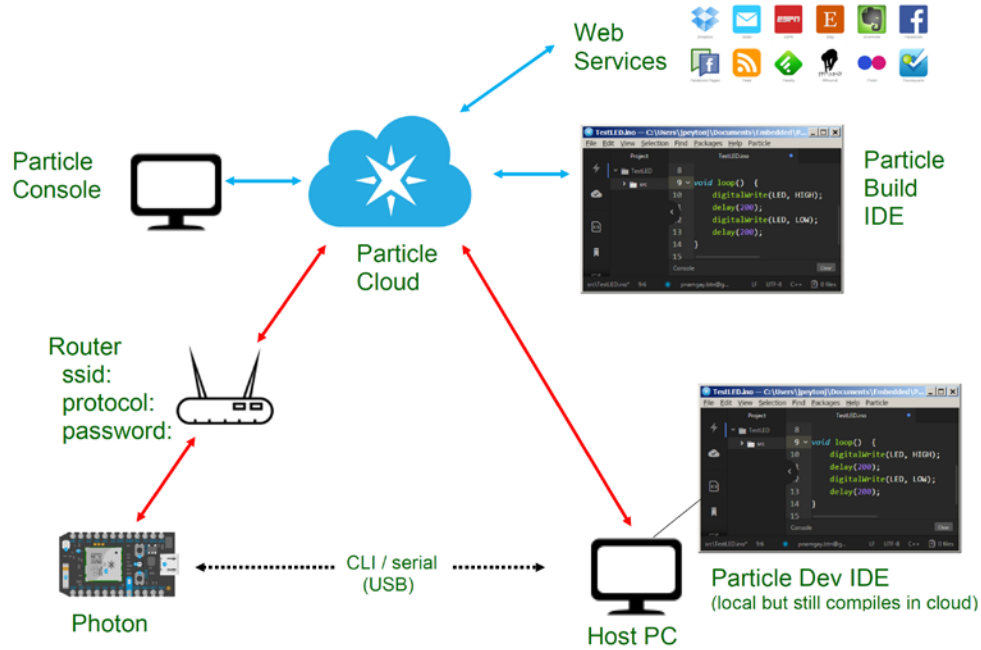


Fig. 1: The Particle world (normal mode)

Lab kit design

As noted in Table 1, several boards come equipped with hardwired sensors or display or multiple LED outputs. Rather than adopt such 'shrink-wrapped' solutions, the aim in this course was to allow students to perform this interfacing themselves using a small, easily portable 3.5" solderless breadboard, Fig. 2. The board brings out 5V and 3.3V supplies to the left and right power rails respectively and partially wires several components for use in the labs: An external LED supplements the onboard LED and highlights the need for a current-limiting resistor. The onboard LED cannot be modulated so the external LED is also used to provide visual feedback (dimming) when studying Pulse-Width-Modulated (PWM) signals. Two external pushbuttons

supplement the onboard pushbuttons and are used to provide digital inputs or to generate edge-triggered hardware interrupts. A Reflective Light Sensor (RLS) acts either as a binary object detection sensor, or as an analog input measurement of reflected light. Analog inputs are also generated using a 10K potentiometer to provide a fraction of the 3.3 V power supply. A transistor together with a flyback diode, provides the capability to switch larger currents (e.g. in order to control a motor) than is possible using the Photon I/O pins directly (even these have a relatively high current drive capability of ± 20 mA). Finally, a piezo buzzer provides auditory feedback for audio-frequency ‘tone’ signals. Each kit contains one such breadboard, together with a collection of jumper cables for making connections, a 5” USB cable (used in this case primarily to provide power to the system), a potentiometer screwdriver, and a small dc motor. The total cost, including the Photon itself is approximately \$30-35.

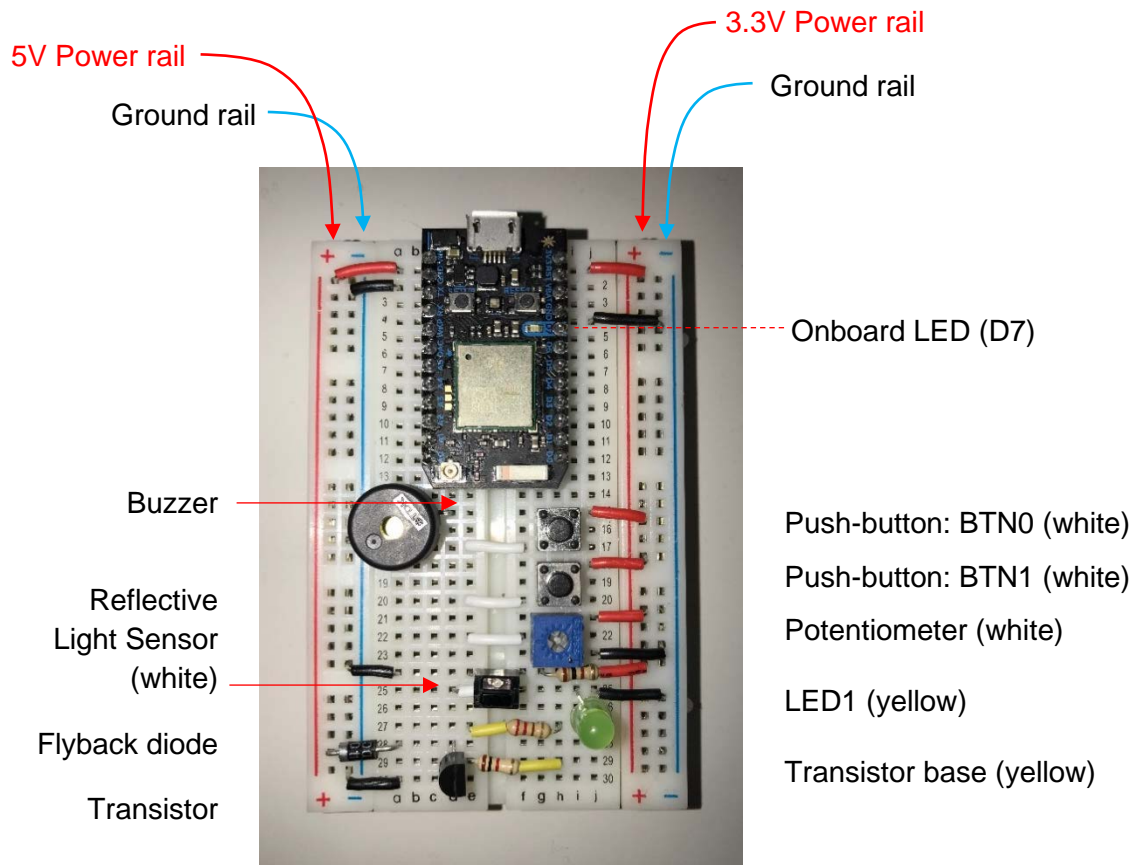


Fig.2: A portable lab kit

Lab Sequence

Simple Digital I/O

The labs build incrementally on one another, starting with the equivalent of a “hello world” program for embedded systems, i.e. a program to flash an onboard LED. Students quickly extend this program to drive an external rather than internal LED. This simple task highlights the relationship between logic levels and physical voltages, and the need for a current limiting resistor to be placed in series with the LED. The logic thresholds for different families of device,

noise margins, and the output drive stage of logic devices are also covered in the classwork for the course.

This material is also relevant in the next step of the assignment where the status of a push-button switch is read by a digital input and used so that the LED flashing now only occurs while the button is pressed. Most of the inputs are 5V tolerant, but the input voltage is undefined (i.e. floating) when the button is inactive. Students are therefore introduced to the concept of pull-up / pull-down resistors and the need to pull the inactive level appropriately. The photon contains internal pull-up / pull-down resistors which can be configured through software – though some care is needed since these resistors are not 5V tolerant.

To reinforce these issues, and to introduce some elementary Boolean algebra, students are asked to modify their program so that flashing occurs if either the pushbutton is pressed OR the Reflective Light Sensor detects an object. The fact that the pushbutton is active high, whereas the RLS is active low means that students need to apply their knowledge in a different context. This also introduces the RLS as a sensor that can be used to detect an intruder in a burglar alarm system which forms part of the final project.

Analog IO

Lab #2 and its subsections focus on analog I/O in contrast to the digital I/O of the previous lab assignment. A/D conversion, covered in class, is used in the lab to read the voltage from the small potentiometer mounted on the protoboard. The potentiometer is connected to one of the 8 A/D converters that are included among the photon's peripherals and used to read a digital value in the range 0 - 4095. Students must be able to compute the corresponding input voltage value, as well as the converter step-size or resolution.

The aim of this lab, however, is to use the potentiometer input to control the brightness of the LED used in the previous lab. Rather than use a true D/A converter, the lab focuses on the more common use of PWM signals where the averaged value yields the output voltage of interest. In this case, the low-level details of setting up timer / counters etc. are hidden beneath an `analogWrite()` function that students call in order to set up the appropriate duty cycle. However, they still need to scale the value obtained from the A/D converter since the duty cycle parameter is an integer in the range 0 - 255.

Continuing to highlight the need to consider physical voltage and current constraints, students are then asked to use the potentiometer to control both the brightness of the external LED, and the speed of a small dc motor included in the kit. The output drive capability ($\pm 20\text{mA}$) of the photon is insufficient for this, so students must use the external transistor (Fig. 2) to switch the current from the 5V supply through the motor. The 'fly-back' diode allows the current in the motor to decay without inducing high voltages as the transistor switches off (Fig. 3).

To provide more insight into different types of digital waveforms, the next lab sub-section asks students to extend their previous work so that as well as controlling the PWM output to the LED, the potentiometer also controls the movement of a small servo motor, and the frequency of the piezo-electric buzzer on their protoboard. The servo signal is generated by instantiating a `Servo` object on an appropriate GPIO pin, and then calling its `write()` function. The buzzer waveform is

generated using the tone() function. In each case the value obtained from the A/D converter must be scaled appropriately before use. Once their programs are working, students are asked to compare and contrast these different waveforms using an oscilloscope. They write a short report on their results, showing how the period / frequency, duty-cycle and pulse-width vary in each case, illustrating their explanations with oscilloscope screen shots and measurements.

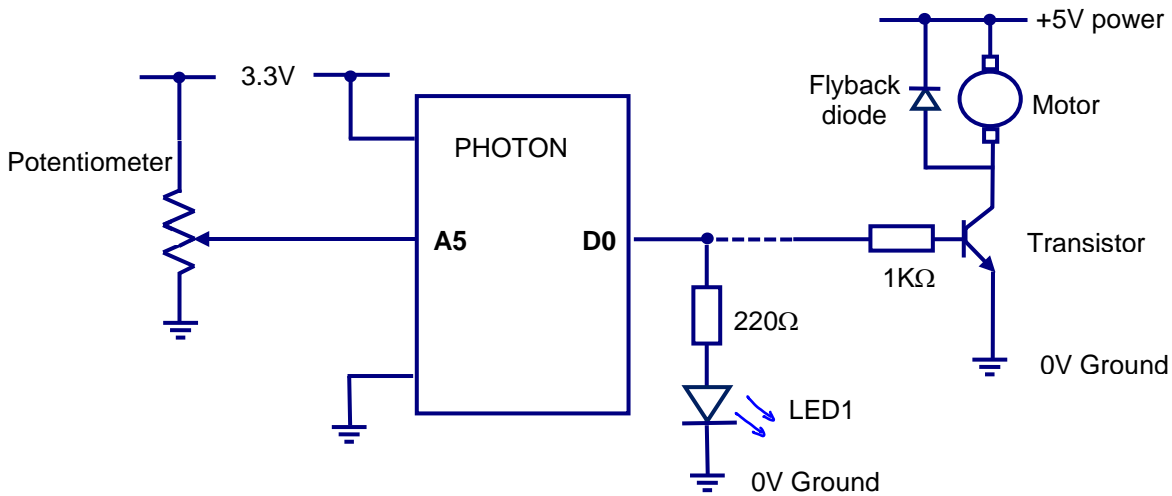


Fig.3: Circuit diagram for Lab #2

Mobile Robot Control

This initial sequence of lab assignments culminates in a small differential drive mobile robot control application, (Fig. 4). In classwork, the use of external transistors to control larger loads such as a dc motor has continued to evolve, including H-bridges for more flexible forward / reverse motor control. In this case, the robot motors are driven by a TB6612 H-bridge driver. Students are required to read the TB6612 datasheet and to figure out how to interface their photon to the robot and how to program the control lines appropriately. The students encapsulate this low-level functionality in a function which sets left and right wheel motor speeds, where positive values indicate forward motion, and negative values indicate reverse. The main program then calls this function in order to drive the robot through a pre-defined sequence of maneuvers. Students test their programs by dropping their protoboard into a tailor-made housing on top of the robot, hooking up the power, ground, and H-bridge control lines and then observing the resulting behavior (Fig.4).

The first robot assignment (above) is typical of open-loop robot control. In the second robot assignment, students program a closed-loop black-line-following strategy using feedback from a downwards-facing reflective light sensor mounted on the robot chassis. The sensor is identical to the RLS device students have already encountered in earlier labs. The RLS is read using one of the A/D converters and compared to the voltage from the potentiometer (measured using another of the A/D converters). If the RLS value is less than potentiometer, then the RLS is deemed to be seeing the white floor. Otherwise, it is considered to be seeing the black line. (The black / white decision threshold can be adjusted using the potentiometer in order to account for any changes in ambient light). The resulting 'error' signal can then be used to steer the robot back towards the black/white line edge so that the robot tracks the line marked on the floor.

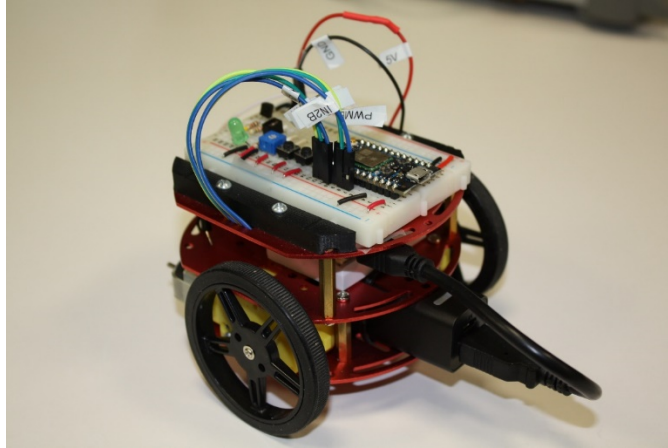


Fig.4: Differential drive mobile robot with downwards-facing RLS sensor

Interrupt Handling

The next phase of the course focuses on events and interrupts. Embedded systems must be able to respond to events that occur in the real world that are unpredictable in the sense that the instant at which they occur is not known at the time the program is written. To emphasize the issues involved, the next lab exercise revisits the flashing LED program. In this case, instead of flashing the LEDs when the button is pressed (and *held* on), the act of momentarily pressing and *releasing* the button should toggle the flashing behavior on or off. Students must therefore detect and respond to the rising edge transition event of the push-button signal.

In the first instance, students are asked to solve this problem using a polling strategy where their program repeatedly checks the status of the push-button signal, responding only when this status changes from a logic 0 to a logic 1. This strategy is awkward to program and consumes CPU attention so that other tasks are neglected. Students are therefore asked to re-design their program in order to attach an interrupt service routine (ISR) to the rising edge of the push-button input signal. The ISR, which simply toggles a flashingEnabled status flag, is then called automatically whenever the event occurs. Students learn about the 'volatile' declaration modifier which is used in this case to alert the compiler that the flashingEnabled variable can be modified at any point and should therefore not be eliminated through compiler code optimization. Meanwhile, the main program is relieved of the need to poll for events and simply flashes the LED's whenever the flashingEnabled variable is true.

Having learned about hardware interrupts, the next assignment explores onboard-timer-generated software interrupts. Up to this point, the 'on' and 'off' periods for LED flashing have been implemented using the delay() function. This means that the CPU cannot simultaneously maintain the flash rate and perform other tasks. The next lab assignment therefore asks the students to extend their program so that the brightness of the external LEDs can be adjusted (using the potentiometer as before) while *simultaneously* flashing the onboard LED at a prescribed rate (assuming flashingEnabled). This is readily achieved using a timer object to generate software interrupts at the desired rate and defining a simple timer ISR to toggle the status of the onboard LED whenever this interrupt is generated.

Communicating with the Cloud

IOT systems are built upon the ability to communicate with the cloud. This can be achieved using one-to-one HTTP requests and responses, or a one-to-many publish / subscribe model. To illustrate the first method, students are asked to expose a variable (in this case containing the ADC value read from their potentiometer) so that it can be monitored remotely from the cloud using a GET request. Building on their experience with interrupt handlers, they also write a small service routine function to respond to PUT requests. In this case, the function toggles the value of the flashingEnabled variable within their program so that the flashing behavior of the system can be modified remotely from anywhere in the world providing the correct security access codes are known. Students test their code by accessing it either through the Particle Cloud console, or from a simple html web page they write containing a single button which initiates the requisite HTTP/PUT request.

To illustrate the second method, students explore the publish/subscribe model by publishing events to the cloud whenever a push-button, connected to their photon, is pressed by the user. Some care is necessary to avoid spamming the Particle Cloud server by publishing events too rapidly (the maximum permissible rate is 1 event/sec or a burst of up to 4 events/sec), but this is unlikely to be exceeded in this case. Students can observe the event that they generate from the Particle Cloud console, but to have it do something more useful they use a web service called IfThisThenThat (IFTTT) to subscribe to the event and respond by triggering a wide variety of possible actions [17]. In this case, students are asked to send themselves a gmail message whenever the event occurs, but it is also simple to send phone notifications, add data to spreadsheets, turn other web-enabled devices on/off etc. More complex cloud programming is possible using node-red [18], a graphically oriented environment for wiring together hardware devices, APIs and online services but time constraints limited this activity.

Local Communication and Serial Protocols

Although communicating with the cloud opens new possibilities for IOT, most communications with other devices occur locally, so it is important to understand how these communications protocols work. The next lab exercise therefore starts by writing a program for serial communication over USB between the photon and a serial monitor on the host PC. At first, the task is simply to echo any messages sent by the host directly back to the host. This proves that communication is working, but the basic program is then extended to echo messages from the PC out to any device connected to the Tx pin of the UART port of the photon. Likewise, any message received on the Rx pin of the UART is echoed up to the host PC. Students can test their program in isolation by looping the Tx output back to the Rx input, in which case the overall effect is still to echo any messages back to the host. However, by pairing with another student, messages can be sent from one host computer, through the UARTs of two connected photon devices, to a second host computer (see Fig. 5).

Having established successful communications in this way, students then explore the signals involved by programming a brief repetitive message and looking at the corresponding waveform on the oscilloscope. Students have to check the baud rate, and decode the message (including start, stop, parity and error bits) and write a brief report of their observations. The mechanics of I2C and SPI protocols are also covered in class, but time did not permit practical implementation in the lab.

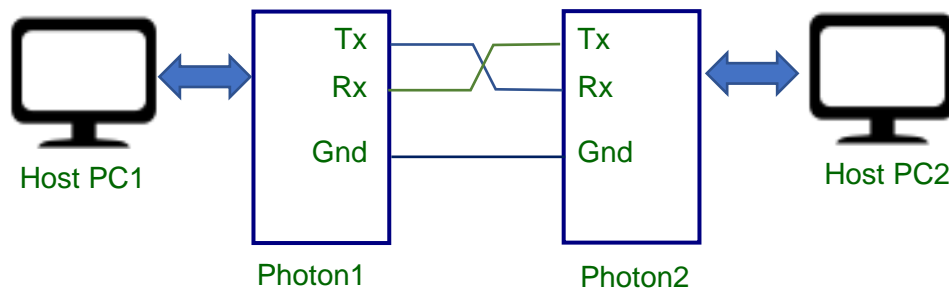


Fig.5: Connecting two student photon boards

Finite State Machines and the Final Project

At this point in the course, students have the ability to handle individual events, whether generated by hardware interrupts, onboard timers, or cloud entities, but they still do not have the skills to detect and respond appropriately to specific *sequences* of events. Such sequential logic is a fundamental topic in digital hardware designs and students are often asked design finite state machines using elementary flip-flops and logic gates, [2],[3]. Software-based finite state machine design is equally important as a programming construct for embedded systems, though this does not seem to be taught as often in EE curricula. One advantage of doing so (in addition to its relevance to embedded systems), is that students can learn concepts and practical application of finite state machines *before* having to implement them in the lower-level world of digital logic. Finite state machines were therefore covered extensively in class and formed a very large part of the final project which is described in more detail below.

The final project brings together everything the students have learned during the semester in an open-ended task that allows all students to demonstrate a baseline of achievement, and stronger students to push the boundaries of their knowledge. The goal is to design a burglar alarm system with a variety of features, using a finite state machine program structure.

The basic system has three states, Off, Armed, and Alarm. The user presses a push-button to Arm the system (in which case the LED shines continuously). Pressing the push-button again disarms it back to the Off state (when all LED's are off). Any button press is also acknowledged by a brief beep on the buzzer. Once Armed, the system responds to intruders detected by the Reflected Light Sensor (RLS) by switching to the Alarm state, during which time the Buzzer alarm is sounded with a tone. The system also alerts the homeowner by sending them an email. The Alarm falls silent (but remains Armed) if the intruder is scared off and departs, or if the system is disarmed back to the Off state.

A variety of additional features can be added to the project. These include delayed Arm activation and delayed Alarm activation so that the homeowner has 30 seconds after arming to leave the building, and 30 seconds to disarm the system when returning to the building – with the new states indicated by flashing LEDs. The alarm itself can also be modified to emit a siren tone rather than the single / pure tone used initially. Students can program a key-code-recognizing finite state machine so that users must press the two pushbuttons in the correct key sequence or

pattern in order to Arm or Disarm the system. Cloud-based functionality can be extended so that users can activate and deactivate the system remotely, (again requiring the correct digital key in order to do so), and so that the system history of events (i.e. the dates and times of activation, deactivation and alarms) is automatically logged to a Google spreadsheet. Students were invited to invent and implement their own features, and some did so – the most imaginative of which had Siri calmly announcing on their smart phones that a burglary was in progress in their homes!

Outcomes & Assessment

Outcome #1: In terms of Bloom’s taxonomy (Fig. 6), the immediate outcome of the lab sequence described in this paper is to deepen the level of knowledge and understanding gained by the students in an associated lecture course by *applying* that knowledge to solve a series of practical problems. At this level, the work contributes to ABET Outcome #1, namely “an ability to identify, formulate, and solve complex engineering problems by *applying* principles of engineering, science, and mathematics”.

The specific application areas covered by the course are summarized in Table 2. Formative assessment of each application is achieved by demonstrating a functioning program and by an evaluation of submitted code according to the following rubric (see later for a summative evaluation):

- **Limited:** The program does not work exactly as specified and is lacking in documentation and clarity.
- **Satisfactory:** The program functions correctly as specified but could be clearer or more elegant; The program is not professionally indented or documented.
- **Mastered:** The program functions correctly as specified; It is well-written and elegant, uses a professional indentation style, and is well-documented.

Assignment	Level	Outcome
a. Simple Digital I/O	Apply	#1
b. Analog I/O (A/D and PWM)	Apply	#1
c. Hardware + Timer Interrupts	Apply	#1
d. Cloud communications	Apply	#1
e. Local / Serial Communication	Apply	#1
f. Finite State Machines	Apply	#1
g. Digital Signals Report	Analyze	#1a,b; #3
h. Serial Signals Report	Analyze	#1e; #3
i. Line-following Robot Project	Create	#1a,b; #4
j. Burglar Alarm Final Project	Create	#1a-f; #4

Table 2: Summary of assignments and corresponding Bloom taxonomy level and Outcomes

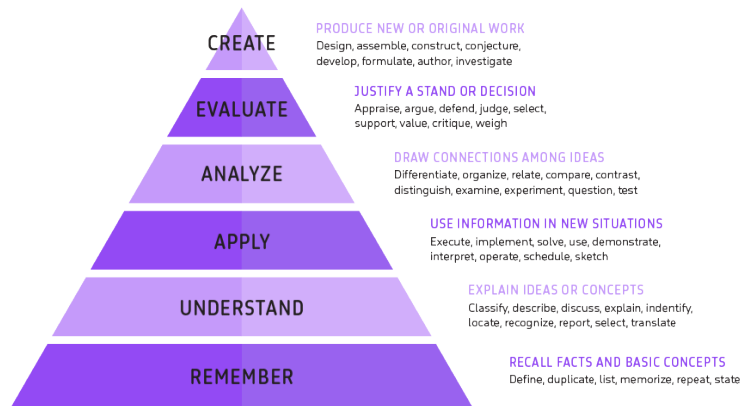


Fig. 6: Bloom's taxonomy pyramid [19]

Outcome #2: A second outcome of the lab sequence addresses higher levels of learning in Bloom's taxonomy is that students should be able to *analyze* and interpret waveforms obtained in selected lab exercises (see Table 2). At this level the work contributes to ABET Outcome #6, namely "an ability to develop and conduct appropriate experimentation, *analyze* and *interpret* data, and use engineering judgment to draw conclusions". Students submit a written report which is assessed using the following rubric:

- **Limited:** Students obtain the correct waveforms but fail to analyze them or derive quantitative measurements from them.
- **Satisfactory:** Students correctly derive quantitative measurements from their waveforms but fail to interpret the meaning of these measurements.
- **Mastered:** Students correctly derive quantitative measurements from their waveforms and provide a meaningful interpretation of the results.

Outcome #3: A third outcome of the course addresses the highest level of learning in Bloom's taxonomy that students are able to build on their skills in order to *create* novel solutions to two open-ended projects as indicated in Table 2. Conveniently, the Line-following Robot project also provides a summative evaluation mechanism for the earlier topics in the course, while the Final Burglar Alarm System project offers a summative evaluation of all the topics in the course. Students submit their code and a written report which is assessed using the following rubric:

- **Limited:** Fails to integrate application skills or to satisfy the base project specification.
- **Satisfactory:** Creatively combines application skills to satisfy the base project specification.
- **Mastered:** Creatively combines application skills to satisfy the base project specification and extends this to include at least five additional creative features.

Outcome #4: A final outcome of the course, as claimed in the introduction, is that students are more motivated and learn more effectively by flipping the traditional bottom-up order of courses in the curriculum. Evaluation of this outcome is still a work in progress, but initial indications are positive. Such evaluation is harder because it entails evaluating different cohorts of students studying under different curricular models. Unfortunately, data was not available for the period prior to the introduction of the new course for EE students, but it is hoped to perform a more

rigorous comparative assessment for CPE students who will take the new course for the first time in 2022. The outcome has two components, one attitudinal (motivation) and one more practical (learning effectiveness). The intention is to work with the Education department to develop a comparative attitudinal survey to assess the first component, and to develop a common practical assessment test given to both cohorts of students to assess the second. For the present, formal evaluation of this outcome remains limited, but subjective course surveys were administered and scored on a scale 1-5, (with 5 being the highest) as further reported below.

Summary of results

The effectiveness of the new lab sequence was measured as outlined in the previous section. A total of 36 students took the course. The majority were sophomore EE students, but the list included five senior ME students, one junior CPE student and two undeclared majors. One student withdrew for personal reasons, but the remainder completed 99% of the assigned work. Aggregated scores (expressed as percentages) are shown in Table 3. The results demonstrate that Outcomes #1, #2, #3 have been met for the large majority (>90%) of the class with significant portions excelling in the ‘Mastered’ category.

As discussed above, the assessment of Outcome #4 is still a work in progress, but initial subjective / survey results are very positive (Table 3). The mean student evaluation obtained for the ‘Overall value of this course’ was 4.8 on the 1-5 scale. Despite the ambitious scope of the work students reported that they ‘Kept up with the assigned work’, again with a mean score of 4.8. Students also noted that the course ‘Encouraged student participation’ (mean 4.9) and was ‘Intellectually stimulating’ (mean 4.5). These figures were in the top quartile for the department. There were relatively few free-style written survey comments, but these included “great job creating these labs”, “Cool labs that are fun to do”, “Fun lab, interesting”, and “I enjoyed being able to apply what we had learned in lecture to make the Photon come to life”. Although these comments are more qualitative and subjective, it is clear that the course was effective in engaging student interest as well as achieving the positive outcomes outlined above.

Outcome	Limited	Satisfactory	Mastered
Outcome #1 (formative)	9%	31%	60%
Outcome #1 (summative)	9%	43%	48%
Outcome #3	9%	54%	37%
Outcome #4	Score <= 3	Score = 4	Score = 5
- <i>Overall value of course</i>	8%	0	92%
- <i>Kept up with assigned work</i>	0	23%	77%
- <i>Encouraged student participation</i>	0	8%	92%
- <i>Intellectually stimulating</i>	8%	38%	54%

Table 3: Summary of assessment results

Conclusions

The design and implementation of a new sophomore-level course in embedded systems and IOT has been described, based on a new low-cost, portable, and Wi-Fi-enabled ARM device. The

hardware kit proved to be robust and flexible, allowing students to explore analog and digital I/O, output pulse modulation schemes, serial protocols, cloud communications, and finite state machine designs. The course was well received, and together both completion and survey data suggest that it succeeded in engaging student interest and learning in embedded systems and IOT.

In future iterations of the course it is hoped to provide more application projects that build upon the current set of exercises. For example, students already use feedback from an RLS sensor to control dc motor speed with an external transistor or H-bridge. A new project is currently being developed where students adapt this program to use feedback from an RLS sensor to suspend a metallic sphere beneath an electromagnet, regulating the coil current using an external power transistor. Another project adds encoders to the mobile robot motors and then uses encoder-driven interrupts to monitor and control motor speed and angular displacements much more accurately than is possible with the current configuration. Yet another project uses a software finite state machine to control a set of traffic lights. It would also be interesting to add an I2C or SPI-driven 4-digit display to the system, and to construct a digital clock / stopwatch / alarm application that can be synced to user's calendars or gps time services. The possibilities go on. Ultimately, the kit proposed in this paper constitutes a very powerful and flexible platform, and one that can help unleash student imagination in the embedded systems & IOT arena.

References

1. *State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating*, IOT Analytics, <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>
2. Clements, A., *The Principles of Computer Hardware*, 4th Edition, Oxford University press, 2006, ISBN: 9780199273133
3. Harris, S.L., Harris, D.M., *Digital Design and Computer Architecture*, ARM Edition, Morgan Kaufmann, 2016, ISBN: 978-0-12-800056-4
4. Orser, D.J., Bazargan, Z., Sartori, J., *Harnessing State of the Art Internet of Things Labs to Motivate First-Year Electrical and Computer Engineering Students*, 2018 ASEE Annual Conference & Exposition, Salt Lake City UT, paper # 22879.
5. Barendt, N., Shridhar, N., Loparo, K., *A New Course for Teaching IoT - a Practical, Hands-On, and Systems-Level Approach*, 2018 ASEE Annual Conference & Exposition, Salt Lake City UT, paper # 22201.
6. Dickerson, S.J., *Preparing Undergraduate Engineering Students for the Internet of Things*, 2016 ASEE Annual Conference & Exposition, New Orleans LA, paper #16609.
7. Mullett, G.J., *Teaching the Internet of Things (IoT) using universally available Raspberry Pi and Arduino platforms*, 2016 ASEE Annual Conference & Exposition, New Orleans LA, paper # 15805.
8. Eke, E.M., *Computer Applications in Mechanical Engineering*, Proceedings of the 2009 ASEE Pacific Southwest Regional Conference, San Diego CA, pp.150-161.
9. California State University, Fresno, *Introduction to Microcontrollers*, https://web.archive.org/web/20110221100533/http://optics.csufresno.edu/~kriehn/teaching/ece1/ece1_files/labs/ece1-lab1.pdf
10. Daugherty, M., *Introducing Programming and Problem Solving with Arduino-based Laboratories*, 2019 ASEE Annual Conference & Exposition, Tampa FL, paper # 26661.

11. Lehman, W., Huang, C-L., Venkatsha, M., Yousuf, A., *Teaching Pic Microcontroller In Eet Program*, 2005 ASEE Annual Conference & Exposition, Portland OR, pp. 10.1224.1 - 10.1224.11.
12. Farook, O., Sekhar, C., Bouktache, E., *Embedded System Design with Microchip's 16F88 Microcontroller*, 2008 ASEE Annual Conference & Exposition, Pittsburgh PA, pp. 13.482.1-13.482.8.
13. "[ArduinoBoardDue](#)". Arduino.cc. [Archived](#) from the original on 2013-01-17. Retrieved 2013-01-18
14. Sentance, S., Waite, J., Hodges, S., MacLeod, E., & Yeomans, L. E. (2017), *Creating Cool Stuff - Pupils' experience of the BBC micro:bit*, Proc. 48th ACM Tech. Symposium on Computer Science Education: SIGCSE 2017 <https://doi.org/10.1145/3017680.3017749>
15. Mudrova, L., Jahoda, V., Porges, O., Krajnik, T., *An omnidirectional mobile robot for large object handling*, Proc. 2011 Research and Education in Robotics-EUROBOT, Prague, Czech Republic, pp.210.
16. Hass, K., Joseph and Juliana Su, *Modernizing the Microcontroller Laboratory with Low-Cost and Open-Source Tools*, 2012 ASEE Annual Conference & Exposition, San Antonio TX, pp. 25.947.1 - 25.947.15
17. If-This-Then-That, Wikipedia, <https://en.wikipedia.org/wiki/IFTTT>
18. Node-Red, website: <https://nodered.org/about/>
19. Bloom's Taxonomy Pyramid, Vanderbilt University Center for Teaching.