# Computer Science Workshop: What Will You Make?

Created by Lyubomir Yanchev on April 8, 2015; last modified by Tom Seaman on April 8, 2015

## Unit Summary

The goal of this workshop is to present the possibilities of Intel® Galileo to students who are already interested in research and innovation. Participants in summer schools, high school robotics clubs, technology classes, and specialized high schools are perhaps best suited. The workshop assumes a certain level of programming abilities and omits the very basics of programming and working with hardware. In exchange, it devotes more time to stimulating students to try things on their own, getting them accustomed to the devices, and testing their affinity for working with open hardware. The underlying concept behind this unit is inquiry-based learning. If adapted to students less experienced with this type of learning, more care should be taken in leading the students through the different exercises. For students with little to no programming experience, this workshop would ideally come after an introduction to programming, or at least should incorporate some introduction to how to use and modify the sample sketches.

## At a Glance

- **Grade**: 9-12
- **Subjects**: Science, Technology & Engineering
- **Topics**: Electronics, Computer Science, Engineering, Design
- **Higher-Order Thinking Skills**: Experimental Inquiry, Analysis
- **Key Learnings**: Electronic circuits, programming, computer communication
- **Content Type**: Unit Plan
- **Time Needed**: Three-day workshop, or 10-15 one-hour class periods
- **Prerequisites**: Some basic programming experience recommended
- **License**: Creative Commons Attribution-ShareAlike (CC BY SA). Read about the license and what you can do with this material here.

## Learning Outcomes

- On the first day, students should gain an understanding of how to connect the Galileo, write sketches, and build circuits on the breadboard
- On the second day students should gain an understanding of the various ways Galileo can interact with other devices, as well as the capabilities of Yocto (Linux)
- On the third day, students should begin to understand the wide range of project opportunities possible with Galileo, and how they might connect to engineering and the natural sciences.

## Things You Need

1. *Intel Galileo*

   Ideally, one board for every two to three students. Even if boards are available for every single student, having them cooperate with one other stimulates the exchange of ideas and encourages them to help one another with some of the tasks. Also, having students work in small groups from the beginning helps the students form partnerships for future projects on day 3. 15-20 students, or 5-7 groups is a good size for the workshop.

2. *Starter Kits*

   There are many good electronics starter kits or Arduino starter kits available. For the open part of the workshop, a large assortment of electronic parts can stimulate creativity and show the versatility of Galileo. Here is an example of a minimal kit.

   1 Breadboard with 840 holes

1 140 jumpers in a box
5 LEDs
1 Button 6x6 mm
1 Photoresistor (LDR)
1 Piezo emitter
1 Spearker 32 ohms
3 Potentiometer 10K ohms
5 Resistors 220 ohms
5 Resistors 10k ohms

3. *MicroSD cards*
   This part is important for working with the Linux capabilities of Intel Galileo. One card per Galileo board, between 1GB and 32GB.

4. *Serial console cable (3.5mm to DB9 RS-232 cable)*

5. *Compatible wifi card (optional)*

## Standards Alignment

This unit is aligned to Common Core National and Next Generation Science Standards.

- Engineering Design: defining engineering problems, developing possible solutions, optimizing the design solution
  - HS-ETS1-1, HS-ETS1-2, HS-ETS1-3
- Physical Sciences: definitions of energy, conservation of energy and energy transfer
  - HS-PS3-1, HS-PS3-2

## Curriculum Framing Questions:

- Essential Question
  Why is experimentation important to developing new ideas and products?

- Unit Question
  How do engineers and computer scientists put experimentation and open inquiry to work to develop new ideas?

- Content Questions
  How can software programs control the behavior electronic components?
  What are the capabilities of the Intel® Galileo board?
  How can Galileo be programmed to interact with its environment?

## Assessment Processes

On day three, students share project ideas and give and receive feedback on their work. Students discuss best practices for working with Galileo.

## Instructional Procedures

**Set the Stage**

Prior to beginning this project, establish an understanding of the basic principles of circuits and programming. Be sure the students understand what a computer program is and what a circuit is, what an LED is, what a resistor is. Discuss how computer programs can be as simple as turning on a light, and as complex as flying an airplane or managing electric power distribution for a large city. Explain that engineers often design a set-up so that they can test their ideas before putting them into practice. Establish a science journal so that students can keep track of key scientific concepts as well as note any questions to revisit.

Invite an electrical engineer or computer scientist to speak with the class about applications of these concepts in the real world.

**Day 1**

*Introduce the Workshop*

Pose the Essential Question, *Why is experimentation important to developing new ideas and products?* Ask students to think individually about the question and then discuss their responses with each other. Ask for volunteers to share responses with the whole class.

Pose the Unit Question, *How do engineers and computer scientists put experimentation and open inquiry to work to develop new ideas?*

Post a chart of the key concepts discussed. Explain to students that they will be exploring these questions throughout their work on the project.

The first day of the workshop concentrates on the basics. The activity starts with the unwrapping, connecting, driver installation, and updating the firmware, instead of starting from the point of a working device. This will enable students to work with the boards alone in the future.

*Part 1: Getting Started*

*Step 1. Explaining the parts*
Start with a complete explanation of the main parts of Galileo (ports, processor, reset, and reboot buttons) and leave the diagram somewhere visible (or dispense printouts), rather than constantly returning to the topic to explain something else. This will be helpful for students who've already had experience with similar hardware, especially Arduino. In general, it won't take more than 15 minutes.

*Step 2: Setting up the Arduino Galileo IDE*
Dispense flash-drives with the installation files or use a local host, instead of relying on internet downloads for each of the participants' computers. This step will usually take at least 15 minutes to complete by everybody, possibly more for some Linux distributions. Latest versions of the software can be found here: https://downloadcenter.intel.com/Detail_Desc.aspx?DwnldID=23171

*Step 3: Connecting the board, installing the drivers and updating*
Walk the students through the installation process, which is different for the different operation systems. It is best to take extra time and cover all of them (you shouldn't have more than three – Linux, Windows and Mac), instead of limiting the students to one OS. Expect this to take 30-45 minutes for everybody, even if the process itself takes much less for one computer. Care should be taken not to accidentally unplug the boards during the firmware update.

*Step 4: Writing sketches*
Students with some basic programming experience should be able understand and write code for Galileo almost immediately. If this is not the case, an introduction to Arduino programming will be needed. Starting with a bare Galileo, show the students the Blink sketch in versions with a delay and an if-else clause. Challenge them to implement different blinking patterns, or give an idea for a Morse code transmitter. Example code with comments is provided here.

This also opens possible discussion for the ways one can communicate with Intel® Galileo, which can lead toward discussions of Linux on the board or serial port communication. One can adapt this step to elaborate on these topics, leave them for later, or give suitable reading materials on them and challenge the students to prepare something before the next day, when these topics will be discussed again. The Morse code transmitter itself is not elegant from a programming point of view, and in order to speed up this step, quicker problems can be given for exercise or discussion. Minimum time one will need is about 15 minutes to explain the basics, and about 30 minutes for the students to play with different blinking patterns or discuss different topics that may arise.

A specific note should be made to students who have experience with Arduino. Unlike Arduino, Galileo doesn't save sketches by default. Here one can cover more about the usage of the SD card, or leave the topic for further discussion the following day.

*Part 2: Working on the breadboard*

Note: Depending on the experience level in the class, this section can be very straightforward, or take a considerable amount of time. It is not uncommon to have a mix of students who have used Arduino for a few years and students

completely new to hardware. This can be handled by pairing the more experienced with the less experienced, or alternatively having experienced and inexperienced groups play with the kits and the board, showing what they can create within an hour or so. Here is a video showing a project by an experienced student in a past workshop who migrated his "piano" from Arduino to Galileo: https://www.youtube.com/watch?v=Up7DPEEINN0&feature=youtu.be

*Step 1: Explain the Breadboard*
Start with explaining the breadboard and the different elements in the kits they have. Resistors and LEDs are most important. A good cautionary demonstration can be the burning of a LED and the reminder that they don't always die quietly, but can instead burst. Then proceed to explain the other elements. Ask the students what parts they would like to use, and if possible make them available for the second day. In this part, the educator should be prepared to explain basic laws of electricity – Ohm's Law, Kirchoff's Law, etc. - if the students have not yet been exposed to them. Depending on the students' preparedness, this step should take about 30-45 minutes.

*Step 2: Recreate the Blink*
Start with asking students to recreate the Blink and its variations on the breadboard. Leave them plenty of time to get accustomed to working with wires, if they aren't already. Remind them to consult with the educators before powering the breadboard to avoid possible problems with short circuits. This should take another 15-30 minutes.

*Step 3: Experiment*
The rest of the time is best left unstructured for open experimentation with the given materials, particularly if the students feel confident using the Galileo board. Alternatively, they can be guided toward the completion of small projects, such as the traffic light that is shown in the code section. Instruct the students to feel free to modify the sample sketches that come with the IDE. If some students have prior experience with Galileo or Arduino, and were given special challenges earlier, ask them to present their creations.

You can maintain the workshop in this mode for as long as the students have the energy to work, but it can easily be 1 hour 30 minutes. One can wrap-up for the day with an explanation of the material that will be covered on Day 3, and an assignment for the students to present an idea for a practical and challenging project.

**Day 2**

The second day of the workshop concentrates on different software aspects of Galileo, primarily serial port communication and the Linux capabilities of the board. The concept of the workshop is to gradually move toward more active student participation. This plan describes several options for the organization of the day. The participation of more than one educator in such event allows some of the activities to be conducted simultaneously.

*Part 1: Serial port communication*
Students who have good programming experience but lack hardware skills naturally orient more toward projects that exploit the communication or analysis capabilities of the devices, and less toward projects which require more electronics skills. Even for those students, however, this is a topic of crucial importance. The basics of serial port communications in some of the most popular programming languages, including C++, C#, and Java can be presented, depending on the preferences of the participants. Stress should fall on the practical applications of serial communication. Prepare to spend 45-60 minutes, depending on the languages.

*Part 2: Using the 3.5mm stereo jack or the USB*
This is considered the most secure way of accessing the terminal. However, it requires some obscure cables - 3.5mm to DB9 RS-232, and then possibly another change of interface to USB. They are not expensive, but they are not always readily available. If this is the case, here is a video showing how students can make their own cable. Making a serial console cable is a very good exercise for students who know the basics of working with hardware, but are not yet confident enough to start making something on their own. It can be time consuming, however. Prepare to spend 60-90 minutes on this.

An alternative to this is using a suitable sketch and a USB connection to the computer. If going this way, take time to explain what the sketch does. This is also a good time to introduce or refresh the students' knowledge of Linux.

*Part 3: Yocto*

It is unlikely that even students with Linux experience will have any experience with Yocto, as this is a seldom-used distribution. Therefore, care should be taken to explain to the participants in the workshop the difference between Yocto and more widespread distributions. Most frequently, the first difference students notice is the need to use opkg instead of apt. One can spend as much time as desired on this topic, but 60-90 minutes is a good estimate. Allow students experienced in Linux to steer the discussion toward the information they need. Explain the difference between the minimal build and the one with the additional SD card. Usually, the questions will revolve around wifi drivers, package compatibility, and programming in languages other than Python and JavaScript. Warn the students that although, bigger distributions such as Debian can in theory be installed on Galileo, their performance is not flawless.

When most of the students are new to Linux, it is best to stick to basic syntax, provide additional reading resources, introduce the package manager, and possibly explain how to enable WiFi on Galileo.

*Part 4: Python*
A basic introduction to Python might be useful, even if somewhat out of the scope of the workshop. Again, this is a topic that can take variable amounts of time, depending on student interest and experience. When working with students who already have programming experience, these explanations can be relatively short.

At the end of the day, lead a discussion of project ideas. Ask the participants to brainstorm a list of their own ideas, and narrow down to one or two candidate ideas that are the most compelling and feasible. Ask the students to turn in their short list of ideas before leaving, but encourage them to continue thinking and brainstorming prior to reconvening for Day 3.

**Day 3**

Day 3 is devoted to the presentation and discussion of project ideas. Ask the students to break out into their small groups once again, and discuss their candidate ideas, adding any new thoughts from overnight. Students should develop some criteria for evaluating their ideas (how compelling is the idea? How feasible? What other criteria can they come up with?). They should use their criteria to choose the best idea they have.

Once they've settled on an idea, they should develop the idea into a presentation. The presentation should cover how they came up with the idea, what problem it will solve, and how they will build it, using the electronics, devices, and software tools they learned about in the workshop.

After a pre-arranged period of time to develop the idea and presentation, perhaps 60-90 minutes, invite each group of students to present their idea in front of the group in a 5-10-minute talk. Encourage group discussion after each presentation. Avoid dampening enthusiasm for ideas that are not feasible; instead suggest simpler objectives or more suitable components, or explain which parts of the project can be done, or transform the idea into something more feasible. Encourage a broad group discussion. Depending on the number of groups, the presentations and discussions could take 1-2 hours.

After the end of the discussion, discuss and list the best practices that came up in the discussion, and suggest some additional project ideas of your own for some class discussion.

Wrap up the workshop by returning to the unit question and essential question. Share your own thoughts about the opportunities available in the vast fields of computer science and hardware engineering. Allow 30-60 minutes for the best practices discussion and wrap-up.

## Prerequisite Skills

- Some basic programming experience is strongly suggested
- Some experience with Arduino or other electronics/hardware device may be helpful

## Differential Instruction

**Students with little to no programming experience**
- Add time for an introduction to how to use and modify the sample sketches

- It is also possible to skip the Yocto and Python sections to spend more time developing Arduino sketches

**Students with programming experience**

- Give students more challenging assignments with electronics and Arduino sketches on Day 1
- Expand coverage of Linux, Yocto, and Python on Day 2
- Have students rapidly their idea or a part of their idea on Day 3

**Students with Arduino experience**

- Students with Arduino experience may difficulty seeing the advantages presented by Galileo. Presenting Galileo as simply "faster and fatter Arduino" misses Galileo's broader advantages. Instead present the Galileo as an important and useful option and challenge the students to create a table, outlining for which type of projects one can stick with Arduino and which can benefit from Galileo in a meaningful way. These tables will give nice discussion opportunities on the third day.

## Additional Background and Tips

The participants in the first implementation of this workshop were the attendees of the 14th annual Research Summer School in Bulgaria (RSS). RSS takes place each August for 40-50 students aged 14-19. It lasts three weeks and is an intensive work environment under the guidance of senior researchers from the Bulgarian Academy of Sciences, doctoral students from leading universities, and representatives of the industry.

Because the participants are self-selecting into this environment, and the educators are technical, the workshop is able to start with the basics and proceed very quickly into advanced material. Participants are expected to be able to assimilate a lot of information, and to already possess significant programming knowledge. Therefore most instruction is aimed at informing and showing, rather than teaching.

Due to the wide number of topics discussed in the workshop, it can be easy for the discussions to get sidetracked into unnecessary minutiae, or for unexpected questions to arise. Questions that cannot be answered immediately are best left for the next day, but some of them can require a fair amount of research on the side of the presenter. We've found one has to be aware of that.


What will you make
Unit Plan 2 Code Ap

**Attachment**: CodeAppendix.pdf