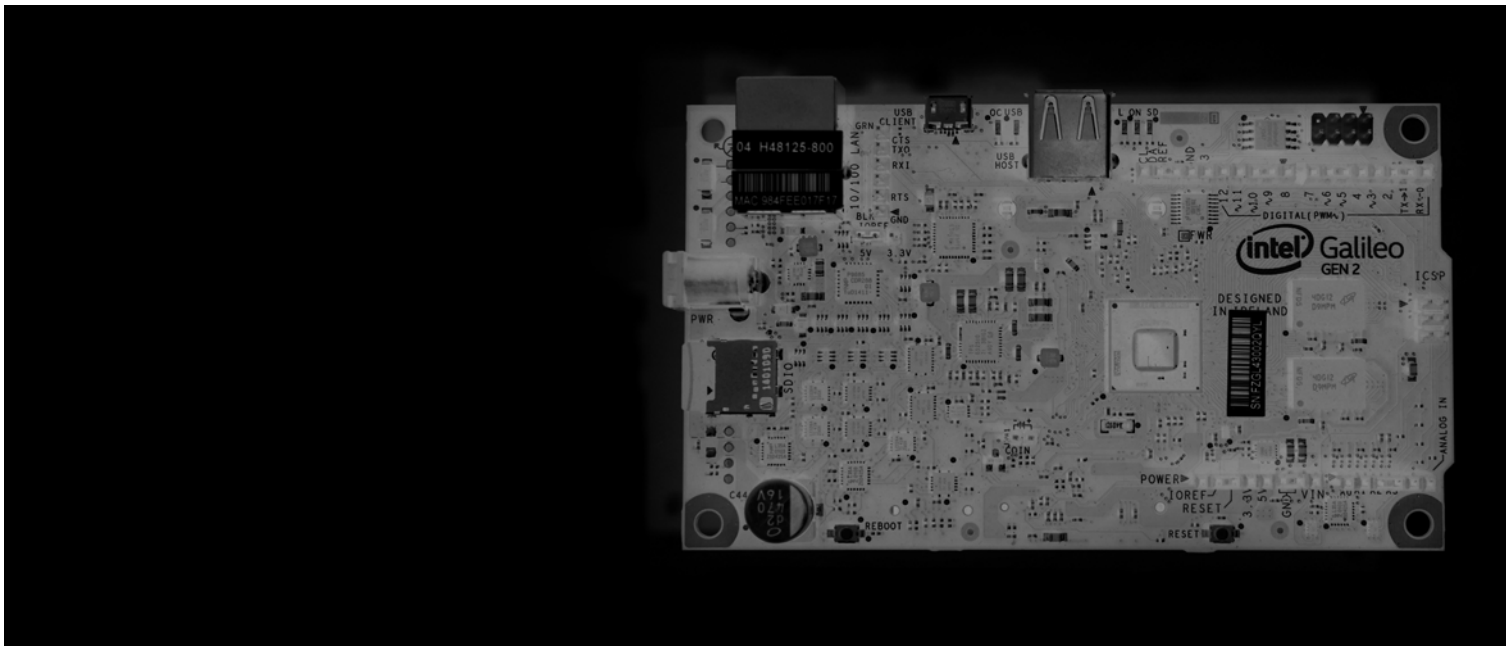
This self-help guide has been developed to provide support in introducing the Intel® Galileo Gen 2 development board into the classroom for learning and teaching.

A Teacher's Guide to the Intel® GALILEO



intel

MacICT
Macquarie ICT Innovations Centre

NSW Education & Communities
Public Schools NSW

MACQUARIE University
SYDNEY · AUSTRALIA

"A Teacher's Guide To The Intel® Galileo"

2015

Macquarie ICT Innovations Centre (MacICT)
Building C5B, Macquarie University
North Ryde, NSW, 2109
(02) 9850 4310

www.macict.edu.au

Authors
Sarah Boyd
John Burfoot
Daniel Green
Cathie Howe

CONTENTS

INTRODUCTION . . . 1
- **Foreword** . . . 1
- **How to use this guide** . . . 3
- **Why use it as a teacher?** . . . 3
- **Linking to the Australian Curriculum** . . . 3
  - Learning Area: Science . . . 3
  - Learning Area: Mathematics . . . 4
  - Learning Area: Technologies . . . 4
  - Learning Area: English . . . 6
  - Learning Area: The Arts . . . 6

PEDAGOGICAL APPROACHES . . . 7
- **Inquiry-Based Learning** . . . 7
- **The 5E Model** . . . 7
- **Project Based Learning** . . . 8
- **Design Based Learning** . . . 8
- **Design Thinking** . . . 8

GETTING STARTED . . . 9
- **What is the Galileo?** . . . 9
- **What is a development board?** . . . 9
- **What is Arduino?** . . . 9

SETTING UP YOUR INTEL® GALILEO GEN 2 . . . 10
- **Getting to know the Hardware** . . . 11
- **Installing the Software** . . . 12
- **Connecting the Board** . . . 13
- **Installing the drivers and other software** . . . 13
- **Launching the Arduino IDE application** . . . 14
- **Updating your board firmware** . . . 14
- **Opening your first sketch: the blink example** . . . 15
- **Select your board** . . . 15
- **Select your serial port** . . . 15
- **Uploading the program** . . . 15
- **Booting from a MicroSD image** . . . 16
  - Which SD Card? . . . 16
  - Formatting your SD Card . . . 16
  - Booting from your SD Card . . . 16

BEGINNING WITH ARDUINO . . . 18
- **Basic Electric Circuits** . . . 18
  - Overview . . . 18
  - Materials . . . 18
  - Resources . . . 18
  - The Blink Example . . . 18
  - Explanation of the Arduino Code . . . 20
- **Visual Programming with Ardublock** . . . 20
  - What is ArduBlock? . . . 20
  - Installing ArduBlock . . . 21
  - Blink with Ardublock . . . 22
  - Button with Ardublock . . . 24
- **Building a Skill Tester: using multiple LEDs** . . . 27
  - Overview . . . 27
  - Materials . . . 27
  - Wiring . . . 27
  - Coding . . . 29
  - Overview . . . 33
  - Materials . . . 33

Hardware construction 34
Wiring 35
Coding 36
Building a night light: using a photocell and LED 39
Overview 39
Materials 39
Hardware construction 40
Wiring 40
Coding 42

USING THE GALILEO AS A COMPUTER 44
Communicating with the Galileo 44
Connecting via Serial 44
Linux and the Galileo 47
Creating a program in Linux using vi 47
Resources 48
Connecting via Ethernet 48
Connecting via SSH 49
Variables 51
Strings 52
Built in function len 52
Modules in python 53
Running Python programs 53
Indentation 54
Javascript and Node.js on the Galileo 54
Overview 54
Materials 54
Coding 54
Node.js Resources: 55
Displaying temperature data on a WebServer using Google Charts 56
Overview 56
Materials 56
Wiring 56
Coding the Arduino 57
Creating the Python Web Server 58
Going Further 60
Resources on Google Charts 60

CONNECTING TO THE OUTSIDE WORLD 61
Sending Email from your Galileo 61
Overview 61
Materials 61
Coding 61
Checking Email from your Galileo 63
Overview 63
Materials 63
Wiring 63
Coding the Python 64
Coding the Arduino 64
Sending texts from your Galileo 67
Overview 67
Materials 67
Wiring 67
Setting up the pushingbox service 68
The Arduino program 72

GLOSSARY 73

ARDUINO REFERENCE 74

INTRODUCTION

Foreword

An education that encourages young people to not merely consume technology, but to understand and create with technology, can change a young person's life.

In a world where technology underpins so much innovation – both in established industries like banking and in emerging industries like additive manufacturing using 3D printing – we want to encourage and inspire the next generation of scientists, inventors and innovators in Australia.

The purpose of this shift in education is to not turn them into an army of app developers – but to equip them with some powerful skills that will allow them to think with technology and at the same time learn valuable skills in science, technology, engineering and mathematics (STEM).

We so often lament the decline of STEM skills amongst Australians and the impact this is cited to have on our economic productivity. But it's not just the responsibility of teachers and governments to encourage this interest. Companies like Intel® can make a positive impact in terms of encouraging young people to be fascinated by the world of science and technology.

Intel® turns sand into computing power. Science, technology, mathematics, engineering and a whole lot of innovation and passion, are at the heart of what we do. And we want to help equip teachers with interesting resources that help them teach STEM skills in the classroom.

Whether it's collecting real time data for a science inquiry project or developing an innovative 'Internet of Things' design to detect pollution in our environment, students now have the opportunity to learn STEM skills in an integrated and engaging way using development boards like Galileo.

Our aim is to arm teachers with some excellent resources to encourage students to both learn STEM but to also experiment, to innovate and to create. Macquarie ICT Innovations Centre is a highly respected provider of ICT professional development for teachers. That's why we approached them to develop this resource - so that it can be a guide for teachers that is:

- A practical, how-to guide for teachers;
- Underpinned by pedagogically sound practices; and
- Aligned to the incoming national curriculum, Digital Technologies.

We are lucky in Australia to have so many inspirational and passionate STEM teachers – and we hope to be able to support and amplify your efforts.

Kate Burleigh
Managing Director
Intel® Australia and New Zealand

The Australian Government's Chief Scientist in his September 2014 report: *Science, Technology, Engineering and Mathematics: Australia's Future*, urges Australians not to be complacent about our place in the global race. The report emphasises the importance for an education in STEM, as the demand for these skills will only grow as we compete in the emerging global economy.

Our education system should support teachers in developing their confidence and ability to teach STEM skills in a learning environment that provides students with the opportunity to engage in active, inquiry-based learning involving technological skills. As students learn how to invent, design and create through projects that involve sustained engagement and collaboration, they will learn more deeply.

Macquarie ICT Innovations Centre (MacICT) welcomed the opportunity Intel® provided us with to develop a teacher's guide to the Galileo. The team spent many hours playing with this development board to determine its potential for learning in an educational context. At MacICT, we are deeply committed to supporting teachers through our professional learning activities, student workshops and research projects to the develop skills necessary to teach Science, Technology, Engineering, Art/Design and Mathematics (STEAM) in the context of the Australian Curriculum. The use of development boards such as the Galileo in student centred learning environments provides students with the opportunity to be designers, problems solvers, collaborators, communicators, innovators and producers. These skills will help prepare them for future life and work. A teacher's skill in designing and managing the learning is integral to the rigour, quality, and success of any project based approach.

This guide has been prepared by teachers for teachers, and is the result of thorough exploration, testing and experimentation by the team who wrote this guide. My sincere thanks goes to Dr Sarah Boyd, Mr John Burfoot and Mr Daniel Green for their enthusiasm and commitment to this project.

Cathie Howe
Centre Manager
Macquarie ICT Innovations Centre

How to use this guide

This workbook is written for all teachers (F-12) to provide support in introducing the Intel® Galileo Gen 2 development board into the classroom for learning and teaching.

It will help guide teachers in designing learning opportunities that develop confidence, creativity and spark an interest in science, technology, engineering, art/design and mathematics (STEAM). No prior knowledge or experience is necessary to complete the projects in this workbook although you may need a strong coffee and some chocolate.

This workbook assumes a teacher is running either a Windows or Mac OSX environment although, the Galileo can be used on a computer running Linux.

Why use it as a teacher?

Using the Intel® Galileo in teaching and learning provides an opportunity for greater student engagement by allowing teachers to create authentic learning projects which can be linked to the real world across several key learning areas, such as the STEAM subject area mentioned above. By their very nature development and prototyping hardware such as the Intel® Galileo, have strong potential for collaboration within the classroom, as students can use these tools to work together to solve everyday design and engineering problems.

Linking to the Australian Curriculum

Using the Intel® Galileo in the context of a student-centred project based approach provides the opportunity for teachers to meet achievement standards across different learning areas including English, Mathematics, Science, Technologies and The Arts. In preparing this guide a specific range of learning areas and achievement standards were targeted in line with the content delivered in the program. The table below highlights standards from different learning areas that may be met through projects using the Intel® Galileo. It is an example only and far from exhaustive.

When using this guide within your classroom, consult the syllabus relevant to your state or territory to best map these achievement standards to outcomes in your context.

Learning Area: Science

Year Level 5-6	CONTENT DESCRIPTION
Science involves testing predictions by gathering data and using evidence to develop explanations of events and phenomena (ACSHE081)	
Decide which variable should be changed and measured in fair tests and accurately observe, measure and record data, using digital technologies as appropriate (ACSIS087)	
Scientific knowledge is used to inform personal and community decisions (ACSHE217)	
Communicate ideas, explanations and processes in a variety of ways, including multi-modal texts (ACSIS093)	
Construct and use a range of representations, including tables and graphs, to represent and describe observations, patterns or relationships in data using digital technologies as appropriate (ACSIS107)	

Year Level 7-10	CONTENT DESCRIPTION
Identify questions and problems that can be investigated scientifically and make predictions based on scientific knowledge (ACSIS124)	
Collaboratively and individually plan and conduct a range of investigation types, including fieldwork and experiments, ensuring safety and ethical guidelines are followed (ACSIS125)	
Collaboratively and individually plan and conduct a range of investigation types, including fieldwork and experiments, ensuring safety and ethical guidelines are followed (ACSIS125)	
Select and use appropriate equipment, including digital technologies, to systematically and accurately collect and record data (ACSIS166)	
Construct and use a range of representations, including graphs, keys and models to represent and analyse patterns or relationships, including using digital technologies as appropriate	
Use scientific knowledge and findings from investigations to evaluate claims (ACSIS132)	
Communicate ideas, findings and solutions to problems using scientific language and representations using digital technologies as appropriate (ACSIS133)	

Learning Area: Mathematics

Year Level 5-6	CONTENT DESCRIPTION
Pose questions and collect categorical or numerical data by observation or survey(ACMSP118)	
Pose questions and collect categorical or numerical data by observation or survey(ACMSP118)	
Interpret and compare a range of data displays, including side-by-side column graphs for two categorical variables (ACMSP147)	
Continue and create sequences involving whole numbers, fractions and decimals. Describe the rule used to create the sequence (ACMNA133)	

Year Level 7-10	CONTENT DESCRIPTION
Identify and investigate issues involving numerical data collected from primary and secondary sources (ACMSP169)	
Explore the practicalities and implications of obtaining data through sampling using a variety of investigative processes (ACMSP206)	
Construct and compare a range of data displays including stem-and-leaf plots and dot plots(ACMSP170)	
Use the language of 'if ....then, 'given', 'of', 'knowing that' to investigate conditional statements and identify common mistakes in interpreting such language (ACMSP247)	

Learning Area: Technologies

Note: At the time of this guide's publication, Technologies was available for use but still awaiting final endorsement.

Digital Technologies - Year Level 5-6	CONTENT DESCRIPTION
Investigate the main components of common digital systems, their basic functions and interactions, and how such digital systems may connect together to form networks to transmit data (ACTDIK014)	
Design a user interface for a digital system, generating and considering alternative designs (ACTDIP018)	

Digital Technologies - Year Level 5-6	CONTENT DESCRIPTION
Implement digital solutions as simple visual programs involving branching, iteration (repetition), and user input (ACTDIP020)	
Manage the creation and communication of ideas and information including online collaborative projects, applying agreed ethical, social and technical protocols (ACTDIP022)	

Digital Technologies - Year Level 7-10	CONTENT DESCRIPTION
Investigate how data are transmitted and secured in wired, wireless and mobile networks, and how the specifications of hardware components impact on network activities (ACTDIK023)	
Investigate how data are transmitted and secured in wired, wireless and mobile networks, and how the specifications of hardware components impact on network activities (ACTDIK023)	
Implement and modify programs with user interfaces involving branching, iteration and functions in a general-purpose programming language (ACTDIP030)	
Plan and manage projects, including tasks, time and other resources required, considering safety and sustainability (ACTDIP033)	
Investigate the role of hardware and software in managing, controlling and securing the movement of and access to data in networked digital systems (ACTDIK034)	

Design and Technologies - Year Level 5-6	CONTENT DESCRIPTION
Investigate how forces or electrical energy can control movement, sound or light in a designed product or system (ACTDEK020)	
Generate, develop, communicate and document design ideas and processes for audiences using appropriate technical terms and graphical representation techniques (ACTDEP025)	
Apply safe procedures when using a variety of materials, components, tools, equipment and techniques to make designed solutions	
Negotiate criteria for success that include consideration of sustainability to evaluate design ideas, processes and solutions (ACTDEP027)	
Develop project plans that include consideration of resources when making designed solutions individually and collaboratively (ACTDEP028)	

Design and Technologies - Year Level 7-10	CONTENT DESCRIPTION
Analyse how motion, force and energy are used to manipulate and control electromechanical systems when designing simple, engineered solutions(ACTDEK031)	
Analyse ways to produce designed solutions through selecting and combining characteristics and properties of materials, systems, components, tools and equipment (ACTDEK034)	
Generate, develop, test and communicate design ideas, plans and processes for various audiences using appropriate technical terms and technologies including graphical representation techniques (ACTDEP036)	
Effectively and safely use a broad range of materials, components, tools, equipment and techniques to make designed solutions (ACTDEP037)	
Develop project plans using digital technologies to plan and manage projects individually and collaboratively taking into consideration time, cost, risk and production processes (ACTDEP052)	
Apply design thinking, creativity, innovation and enterprise skills to develop, modify and communicate design ideas of increasing sophistication (ACTDEP049)	

Learning Area: English

Year Level 5-6	CONTENT DESCRIPTION
Understand how texts vary in purpose, structure and topic as well as the degree of formality(ACELA1504)	
Navigate and read texts for specific purposes applying appropriate text processing strategies, for example predicting and confirming, monitoring meaning, skimming and scanning(ACELY1702)	
Plan, draft and publish imaginative, informative and persuasive texts, choosing and experimenting with text structures, language features, images and digital resources appropriate to purpose and audience(ACELY1714)	
Use a range of software, including word processing programs, learning new functions as required to create texts (ACELY1717)	

Year Level 7-10	CONTENT DESCRIPTION
Use a range of software, including word processing programs, to confidently create, edit and publish written and multimodal texts(ACELY1728)	
Plan, draft and publish imaginative, informative and persuasive texts, selecting aspects of subject matter and particular language, visual, and audio features to convey information and ideas (ACELY1725)	

Learning Area: The Arts

Note: At the time of this guide's publication, The Arts was available for use but still awaiting final endorsement.

Year Level 5-6	CONTENT DESCRIPTION
Develop and apply techniques and processes when making their artworks (ACAVAM115)	
Plan the display of artworks to enhance their meaning for an audience (ACAVAM116)	

Year Level 7-10	CONTENT DESCRIPTION
Develop ways to enhance their intentions as artists through exploration of how artists use materials, techniques, technologies and processes(ACAVAM119)	
Manipulate materials, techniques, technologies and processes to develop and represent their own artistic intentions (ACAVAM126)	

PEDAGOGICAL APPROACHES

As teachers, we know that quality classroom teaching has a profound influence on student learning. Just as consideration is given to **what** we want students to learn, equal consideration needs to be given to **how** we want students to learn, that is what pedagogies we may employ to best suit the learners and learning context.

In consideration of the broad range of learning contexts that could benefit from integration of the Galileo into students' learning, we have utilised different pedagogies and models in the design of activities within this workbook.

Both through the Arduino environment and the use of development boards such as the Intel® Galileo, students are able to explore ways that programming can affect the physical world. Collaborative activities and experimentation enable students to find new ways to engage with curriculum material.

We have suggested some models below as a starting point, but as you begin to imagine other creative projects for your students, you may employ new pedagogies.

Inquiry-Based Learning

Inquiry-Based Learning (IBL) is a constructivist pedagogy which supports a student centred, authentic and often ubiquitous (anywhere, anyhow) learning. Inquiry-based learning and teaching is a feature of the Australian Curriculum in a number of learning areas and is integral in the Critical and Creative Thinking General Capability.

Within inquiry-based learning an emphasis is placed on evidence; when provided a problem, students use the information at hand to construct logically-sound questions which then fuels further investigation. These inquiries can be conducted individually, or in pairs and groups. As students communicate and compare their findings, they must use their collected evidence to justify and evaluate their results, which in turn can spark new lines of inquiry.

The 5E Model

The 5E's (Bybee 2006) is an instructional model of implementing inquiry-based learning within classrooms and is a good reference point for teachers to develop a deeper understanding of what constitutes quality teacher practice. It uses a specific sequence of inquiry to assist students in working through syllabus materials:

- **Engage**
- **Explore**
- **Explain**
- **Elaborate**
- **Evaluate**

Engaging students involves knowing your students, their interests, assessing their prior knowledge and introducing problems that promote curiosity. The **Explore** stage encourages students' innate sense of discovery, providing an environment where they can take risks and involve the expertise of all students. This stage provides students with a common base of activities that draws upon prior knowledge and the generation of new ideas. Students have the opportunity to explore and design.

This stage provides students with the opportunity to **Explain** their findings based on their collected evidence. From here, a new concept, process or skill may be introduced by the teacher and further explicit instruction may be issued along with clarification of any relevant terminology. This may help guide students towards a deeper underdtanding.

The **Elaborate** stage challenges students to extend the understanding and skills through new experiences. Students apply newfound information to their learning, and explore the potential to expand their knowledge for other learning areas. Finally, through **Evaluating** students can reinforce their understanding and abilities by providing opportunities to reflect on their work; assisted by open-ended questions and feedback from teachers.

Though the 5E Instructional Model has found more widespread use within science classrooms, there are ample opportunities for these practices to be rolled out in the teaching of coding, programming and making. At its heart, programming is a process of problem solving and experimentation; using code to find the most efficient way answer to a problem or need of a user. More information about this approach can be found here: http://www.tale.edu.au/tale/live/teachers/shared/BC/science_inquiry_approach.pdf

Project Based Learning

Project Based Learning (PBL) is a student-centred teaching method in which students gain knowledge and skills by working for an extended period of time to investigate and respond to a complex question, problem, or challenge. (Buck Institute for Education: http://bie.org) PBL is designed to encourage deep levels of understanding by applying knowledge to real-world problems and situations.

Within a PBL model, core curriculum elements and key concepts are covered in-depth within lessons, but this content is then reinforced through dedicated time spent solving a single problem in great depth. It a strongly collaborative approach, and lends itself very well to cross-curricular engagement. Tasks are issued to students through open-ended "driving questions" that are formulated in such a way as to spark discussion and encourage multiple approaches to the task.

Design Based Learning

Design-based learning (DBL) is a form of project-based learning in which involves the students in the process of developing, building and evaluating a product they have designed (Silk 2009). Working and completing design based activities can make students feel proud of their achievements, as well as building up their confidence as thinkers, designers, and doers that will benefit them through their education and life (Barron et al. 1998). "Most classroom problem-solving activities focus on analytic thinking: decomposing problems into sub-problems. Students rarely get the opportunity to design and invent things" (Resnick 1990). Learning through design encourages students to think and act creatively, to generate new ideas, design prototypes, make improvements and create final products. This type of learning often involves playful experimentation – trying new things, tinkering, testing, taking risks and iterating over and over.

Design Thinking

Similarly to Project Based Learning, Design Thinking is a process that involves examining and deconstructing problems and tasks before investigating the best way they can be solved.

As a process Design Thinking is commonly used within the commercial sector, but has recently begun to find traction as a way to engage students. It stems from a belief that people of all backgrounds can express creativity when provided with the right foundations. In this case, it begins with the question itself.

Like the 5E Model, Design Thinking incorporates a series of stages that problems can be worked through. There are several models that can be used, but many consist of the following ideas:

- *Define the problem*: Break down the task into its core components, and determine what exactly is the problem that needs to be solved
- *Interpret the problem, and create new ideas:* Find different ways to approach the task, consult different opinions and points of view, and start to follow interesting leads
- *Execute the solution:* From the ideas generated, start to pursue the strongest methods to fulfil the needs of the task. This often involves the creation of a prototype
- *Refine the solution:* Take your finished work, and conceive ways it could be better. This in itself may require another iteration of Design Thinking.

As a platform for using development boards and prototyping hardware within a classroom, strategies like project based learning and design thinking are an ideal way of taking an abstract concept like coding and programming, and making this work tangible for students. The sample projects contained within this workbook will demonstrate how the Intel® Galileo can bring the gap between learning in the digital and physical worlds.

GETTING STARTED

What is the Galileo?

The Intel® Galileo is a development board which is hardware and software compatible with the Arduino platform. As well as encompassing a range of Arduino's capabilities, the Galileo also runs an open source Linux operating system and can be connected to the internet via Ethernet or Wifi.

What is a development board?

A development board is a piece of hardware, featuring a microcontroller built on to a single printed circuit board. This hardware allows users to create devices that interact with their environment using input sensors and output actuators. After being programmed by the user microcontrollers can perform specific tasks, depending on the capabilities of the board itself. Some development boards also include microprocessors so they are effectively small computers without the peripherals.

Because of their small size and relatively low-cost, development boards such as the Intel® Galileo and Arduino Uno have grown in popularity in recent years. They allow users of all levels of experience the chance to understand electronics in a physical capacity, from there tinker and experiment with making their own devices.

What is Arduino?

When talking about Arduino as a platform, it is important to understand that this refers to both hardware *and* software.

Arduino *hardware* is a series of development boards that come in a variety of models depending on the needs of the user. Types of "official" Arduino boards include the Arduino Uno, Arduino Mega and Arduino Leonardo.

There are also "Arduino-Compatible" boards, that share the capabilities of "official" Arduino boards, but often feature expanded functionality. The Intel® Galileo is an Arduino-compatible development board, sharing some features and capabilities of the Arduino Uno R3 development board.

Arduino's *software* refers to the Arduino Integrated Development Enivronment (IDE), a computer program used to load programs onto an Arduino development board. Users write programs for Arduino hardware using an amalgamated version of the C and C++ programming languages. This language is also referred to as *Arduino*.

The Arduino IDE runs on Windows, OSX and Linux machines. The programs can be downloaded using a cable (such as USB) which connects the Arduino hardware to your computer

The basic functions of the Intel® Galileo, like the Arduino, are to read values from a set of hardware inputs and activate a set of hardware outputs using programs coded in the Arduino programming environment. Typical inputs the Galileo can read include switches, temperature sensors and buttons while typical outputs include LEDs, motors and text displays.

SETTING UP YOUR INTEL® GALILEO GEN 2

Your Intel® Galileo kit contains:

- 1x Intel® Galileo Gen 2 development board
- 1x 12V power supply with cable

Note: ***You must connect the power supply to the Galileo first before connecting the USB cable. You will damage the board if you power it with a USB cable.***

WARNING: The power supplies are NOT interchangeable between Intel® Galileo and Intel® Galileo Gen 2 boards. The Intel® Galileo Gen 2 power supply is a higher voltage (12V) and it will permanently damage the first generation Galileo boards.

Figure 1 Micro USB cable for downloading programs to your Galileo

Getting to know the Hardware

Figure 2 Intel® Galileo Gen 2 Front View

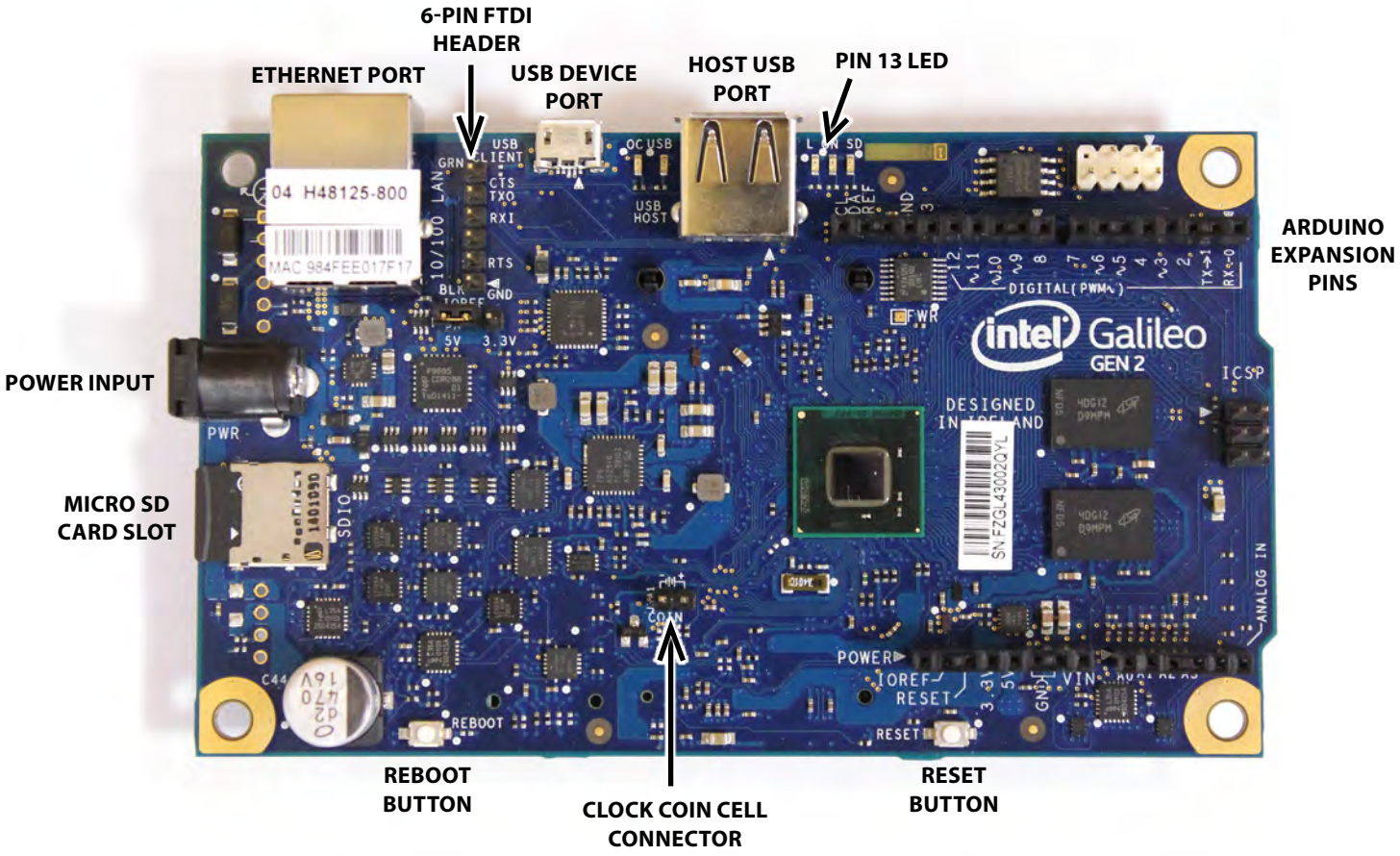
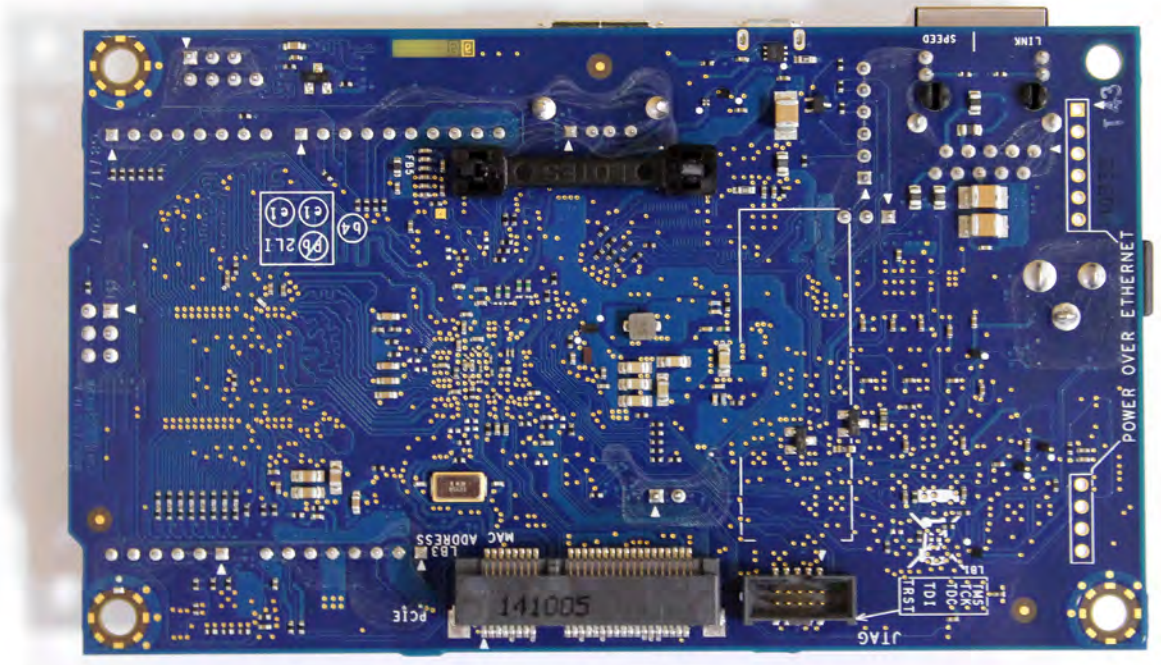


Figure 3 Intel® Galileo Gen 2 Back View



Here is a brief description of some of the elements on board the Galileo Gen 2. For a detailed schematic diagram of the board go to www.intel.com/content/www/us/en/embedded/products/galileo/galileo-overview.html

MICRO SD CARD SLOT

This is where you slot in the micro SD card which allows the Galileo to run a more powerful version of Linux.

ARDUINO EXPANSION PINS

These pins allow you to connect inputs and outputs to the Galileo. They are compatible with Arduino shields allowing them to plug straight in.

USB DEVICE PORT

This is where you plug in the USB cable for downloading Arduino programs to the Galileo from your computer.

HOST USB PORT

This is where you can plug in external peripheral devices such as webcams, sound devices etc..

6-PIN FTDI HEADER

This is where you can plug in a serial cable to communicate with the Galileo Linux operating system serially from your computer. This is a TTL level interface which means it cannot connect directly with your computer serial port. The cable supplied by Intel® includes a TTL -> USB adapter.

POWER INPUT

This is where you plug the 12 V power supply in. You must always power the board before connecting it via USB to your computer otherwise you may damage the board.

ETHERNET PORT

This allows you to connect the Galileo to the internet via an Ethernet cable.

MINI PCI EXPRESS SLOT

This is located on the back of the board and is where you can plug in your WiFi card.

CLOCK BATTERY POWER

You can plug in a small 3V coin battery here to allow the Galileo to retain the date and time when the power is not connected. It does not recharge the board.

REBOOT BUTTON

This button will reboot the board including Galileo's Linux operating system.

RESET BUTTON

This button will restart your code/sketch and send the reset signal to any shield attached to the expansion header. The Linux operating system will be unaffected.

Installing the Software

The Arduino IDE for Intel® Galileo is available here: https://communities.intel.com/community/makers/drivers You MUST download this version of the IDE, as it has been specially designed to work with the Galileo's Quark processor. You CANNOT use the regular Arduino IDE software as it will not recognise the Galileo board.

You will need up to 200 MB of free space, depending on which operating system you are using.

Download the zip file for your operating system labelled 'Arduino Software 1.5.3'. At this point you do not need to download any of the other files from this web page.

SETTING UP ON WINDOWS

1. Right click on the file you downloaded and select 'Extract All'. Select the destination to extract the file into, e.g the C:\ directory or the desktop.

Alternatively, if you have a third party unzip tool like 7-zip from http://www.7-zip.org you may use this to do the extraction.

SETTING UP ON MAC

1. In the Finder, double-click the zip file to decompress the IDE application.
2. Drag and drop the Arduino application onto the Applications folder on your Mac.

Connecting the Board

Connect the 12 V power cable to the Intel® Galileo Gen 2 board and to a power outlet. The Power LED shown above (board label = ON) will turn on.

Note: ***Always connect the power before the USB connection.***

Installing the drivers and other software

WINDOWS:

1. Connect the USB cable to the USB Client Port (closest to the Ethernet) and to a PC. Wait for Windows to begin its driver installation process. After a few moments, the process will fail.
2. Click on the Start Menu, and open up the Control Panel. While in the Control Panel, navigate to System. Next, click on System. Once the System window is up, open the Device Manager. You may get a pop up box telling you that you cannot make changes. Just click on OK.
3. Look under Ports (COM & LPT). You should see an open port named Gadget Serial V2.4. If you do not see this open port, follow steps a-c in the Note below.
4. Right-click on the Gadget Serial V2.4 port and choose the Update Driver Software option.
5. Choose the Browse my computer for Driver software option.
6. Navigate to the hardware/arduino/x86/tools directory. This allows the proper driver file linux- cdc-acm.inf to be installed.
7. Once the driver is successfully installed, Device Manager will show a Galileo (COMx) device under Ports (COM & LPT). Note the COMx port number as it will be needed in the IDE later. The example below shows COM5.

Note: ***You may find Gadget Serial 2.4 under Other devices in Device Manager. If so, do the following:***

8. In Windows Explorer, go to C:\Windows\System32\drivers and look for usbser.sys. If it is present, you can skip these steps. If it is missing, copy usbser.sys from the archive location identified below:

   Windows 7: C:\Windows\System32\DriverStore\FileRepository\mdmcpq.inf_amd64_neutral_fbc4a14a6a13d0c8\usbser.sys (archive file)

   Windows 8: C:\Windows\System32\DriverStore\FileRepository\mdmcpq.inf_amd64_d9e0b9c4fe044b4d\usbser.sys (archive file)

9. In Windows Explorer, copy to: C:\Windows\System32\drivers. You may need to provide Administrator Permission to complete the copy.
10. Once usbser.sys is copied, continue with step 3 above.

MAC OS X:

The board is supported by the OS X built-in USB drivers, however, the board has to boot for it to show up because the port is driven by software on the board.

1. If not done already, connect the power cable to the board and to a power outlet. Wait for the board to boot.

Note: **Always *connect the power before the USB connection.***

2. Connect the USB cable to the USB Client Port (closest to the Ethernet) and to your Mac. Check the System Profiler > USB setting to be sure that Gadget Serial is selected. If you are installing a new version of the IDE, you may need to re-select this setting.
3. In the Arduino IDE, the correct serial port shows in the Tools > Serial Port menu as /dev/cu.usbmodem*nnnnn* where *nnnnn* is a number such as fd121. Do not select the /dev/tty port.

Launching the Arduino IDE application

WINDOWS:

In the folder arduino-1.5.3, double-click arduino.exe

MAC OS X:

Open the Launchpad from the dock and click on the Arduino application icon.

Updating your board firmware

The IDE contains the release-specific firmware for your board. Follow the steps below to update your board firmware using the IDE.

1. Remove all power from the board (USB and power cord). This makes sure that no sketch is running on the board.
2. Remove the SD card from the board (if it is inserted).
3. Power up the board by plugging in the power supply.
4. Connect the USB cable to the USB Client Port (closest to the Ethernet). Note which COM port it is connected on.
5. Launch the IDE and select the board via Tools > Board > Intel Galileo or Tools > Board > Intel Galileo (Gen 2).

6. Select the correct serial port using **Tools > Serial Port.**

Note: ***Do not download any sketch to the board before you upgrade the firmware.***

7. Launch the software upgrade using **Help > Firmware Upgrade.**
8. A message is displayed asking you to confirm that the power cable is plugged in. Click Yes if it is connected. If no cable is plugged in, exit the upgrade process by selecting No, connect the power, and restart this process.
9. The board can be upgraded to newer software or downgraded to older software. The next message displays the current software version that is on the board and the software version that you are trying to flash onto the board. Select Yes to either Upgrade/ Downgrade or flash the same software again.
10. The upgrade progress takes about 6 minutes and is displayed in several popup messages. During the upgrade process, you will not have access to the IDE.

Note: ***The power and USB cables must stay connected during the upgrade process.***

11. When the upgrade completes, a message is displayed stating **Target Firmware upgraded successfully**. Click **OK** to close the message.

Note: ***On OS X, you must reboot the IDE before continuing.***

Opening your first sketch: the blink example

In the Galileo, programs are called sketches. A range of example programs are provided within the software. The most basic one is called Blink and it makes a light blink. Open the LED blink example sketch from the IDE by going: **File > Examples > 1.Basics > Blink.**

Select your board

Select **Tools > Board > Intel Galileo (Gen 2)**.

Note: ***If the IDE was closed and then reopened, it will default to a board that was previously selected. You may need to explicitly select your board.***

Select your serial port

Select the serial device of your board from the **Tools > Serial Port** menu.

WINDOWS

Use the *COMx* number assigned earlier. You can retrieve the port number by navigating to:

Start > Control Panel > System and Security > System > Device Manager.

Look under **Ports (COM & LPT)** to see which *COMx* is assigned to **Gadget Serial**.

MAC OS X

Use /dev/cu.usbmodem*nnnnn*

Uploading the program

Click the **Upload** button in the IDE and wait a few seconds. If the upload is successful, the message **Done uploading.** will appear in the status bar.

A few seconds after the upload finishes, you will see the Pin 13 LED on the board (shown above) start to blink. Congratulations! You've gotten your board up and running.

Booting from a MicroSD image

The Intel® Galileo Gen 2 out of the box comes with a very small Linux image on board the flash storage which has limited storage and capabilities. However, you can boot your board from a micro SD card with a larger Linux image, enabling you access to many more Linux utilities and extra storage.

You need to boot from the SD card if you want to use WiFi as the WiFi driver cannot fit on the onboard flash storage. The Linux SD image also includes several linux utilities such as ALSA, V4L2, python, SSH, node.js, and openCV.

Which SD Card?

Your micro SD card can be any size from 1 to 32 GB and SDHC format. SDXC format is **not** supported. You will need an SD Adapter so your micro SD card can fit in your SD slot of your computer.

Formatting your SD Card

Your SD card must be formatted as FAT or FAT32. Fortunately, most SD cards you buy are formatted as either FAT or FAT32. In order to format an SD card as FAT do the following:

Place the micro SD card inside your adapter and place your Adapter in your SD card slot of your computer.

WINDOWS

1. Click the Start menu icon and click "Computer".
2. Right click on the SD card drive under the heading Computer.
3. Select 'Format' and you will be provided with a pop up dialog.
4. Now select FAT to begin formatting your SD card.

MAC OS X

1. Click Applications.
2. Double-click the Utilities folder within Applications. Double click the Disk Utility app icon to launch the application.
3. Select your SD card on the left side of the Disk Utility program. Click the Erase tab.
4. Click the drop down box next to 'Volume Format' and select FAT.
5. Type in a name for the SD card next to "Name". Click the "Erase' button to begin formatting the SD card. Wait for the 'Format Complete' box to appear before disconnecting the SD card from your Mac.

Note: ***You will lose any data that is on the SD card when you do this formatting.***

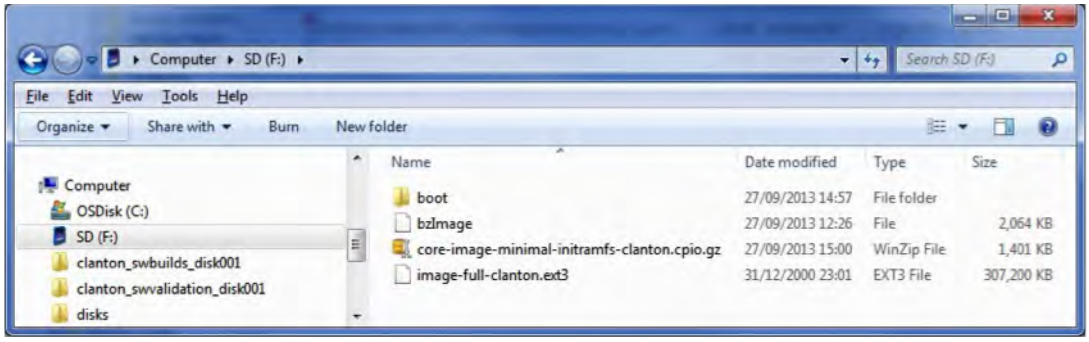
Booting from your SD Card

Follow these steps to boot your board from an SD card. This setup also allows you to save your sketch to the board, so it will be able to repeat sketches after board power-down. (These steps create a persistent /sketch folder and rootfs (root file system).

1. Download the SD-Card Linux Image file from the following website:

   https://communities.intel.com/community/makers/drivers

2. Copy all files and directories from the zip file to your SD card.



Note: ***You do not need to create a directory on the SD card. The zipfile contains all the necessary files and structure. Be sure it is extracted at the top level of the SD card. See Figure 3 for a sample view.***

3. Insert the micro SD card into the slot on your Intel® Galileo, then power on the board.

Note: ***The first time you boot the board may take several minutes. This is expected behavior due to the SSH component creating cryptographic keys on the first boot.***

Now you have access to a larger Linux image with many more Linux utilities and the ability to connect to WiFi.

BEGINNING WITH ARDUINO

Basic Electric Circuits

Overview

In this activity, students will connect up the Intel® Galileo to a simple circuit and make an LED blink using Arduino code.

Materials

- LED
- Breadboard
- Intel® Galileo board
- Intel® Galileo software
- 100 Ω Resistor

Resources

http://www.scootle.edu.au/ec/viewing/L3059/index.html
http://www.australiancurriculum.edu.au/Elements/ACSSU097
http://www.youtube.com/watch?v=MBRTR2dlwvA&cc=1
http://www.teachingideas.co.uk/science/contents_circuits.htm
http://www.scootle.edu.au/ec/p/accessing_scootle

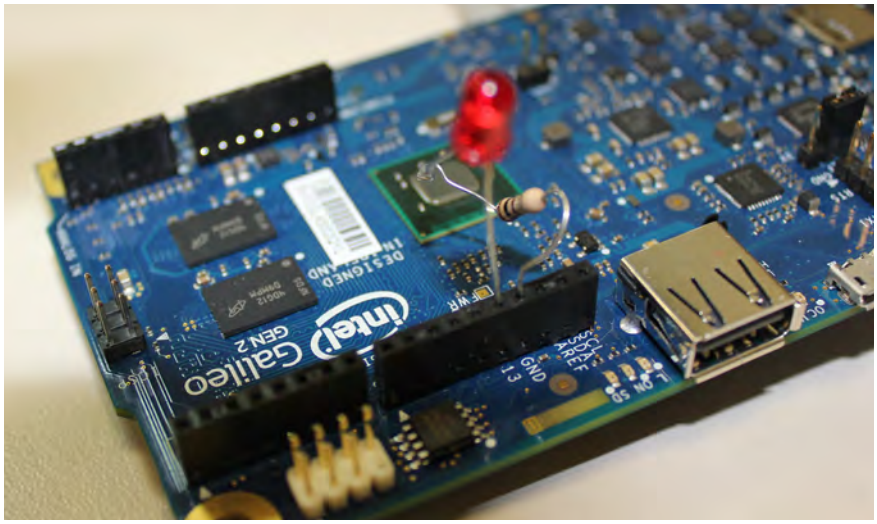
The Blink Example

WIRING INSTRUCTIONS

LEDs have a negative and a positive lead and current only flows one way. The convention is that the shorter lead is the negative lead while the longer lead is the positive lead.

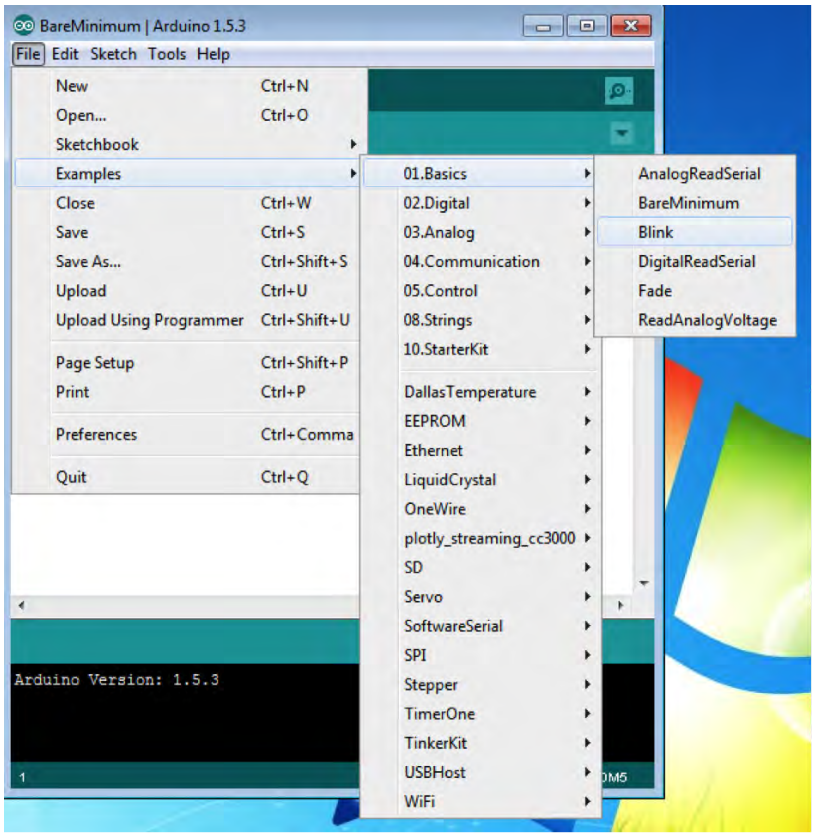
Connect the negative (shorter) lead of the LED to a resistor by twisting them together and connect the other end of the resistor to GND next to Pin 13.

Connect the positive (longer) lead of the LED to Pin 13. Make sure there is no touching between the resistor and the positive lead of the LED.



CODING INSTRUCTIONS

From the Arduino IDE, go to File > Examples > 01.Basics > Blink and open the example code.




```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
 */


// Pin 13 has an LED connected on most Arduino boards.
int led = 13; // this assigns an integer type variable named led to pin 13

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Explanation of the Arduino Code

In Arduino programs, there are always two functions: setup and loop.

The setup() function runs once when you load a new program or press the reset button. It is represented by the code:

```
void setup() {
  ….
}
```

The loop() function runs after the setup function and runs over and over again forever. It is represented by the code:

```
void loop() {
  ….
}
```

The digital pins on the Arduino, (0 – 13) located next to the USB slot can be set to be either output or input pins using the command, pinMode(). The pinMode function expects two inputs inside the parentheses: the first input is the number of the Arduino pin while the second input is either OUTPUT or INPUT.

In order to set an output on the Arduino we use the command digitalWrite. It also takes two inputs: the first input is the number of the Arduino pin to write to and the second input is whether to write HIGH or LOW.

The delay command makes the program wait for a certain number of milliseconds before moving onto the next command. Thus:

```
delay(1000);
```

instructs the code to wait for 1 second.

A NOTE ON COMMENTS

Comments are lines of script inside an Arduino sketch that provide notes to yourself or other users. These can be about the program's functionality, any issues with the code, or anything else you would like a user to understand about your program. Comments are ignored by the Arduino IDE and are not uploaded to the Intel Galileo hardware. Comments are often signified with the symbols "//" or "/* */" and can be written as follows:

```
// This is a single line comment. Anything after the slashes is a comment
// to the end of the line
/* this is a multiline comment
- use it to comment out whole blocks of code
*/
```

Visual Programming with Ardublock

What is ArduBlock?

ArduBlock is an optional add-on to the Arduino IDE that uses a visual programming language to generate code for Arduino hardware.

Using an interface similar to other visual programming languages such as Scratch, Blockly, and Hopscotch, ArduBlock can make programming Arduino hardware more accessible for younger students or provide a quick solution to testing new programs. After creating a program in ArduBlock, students can export the code as C/C++ for use within the Arduino IDE, allow them to transition to text-based coding.

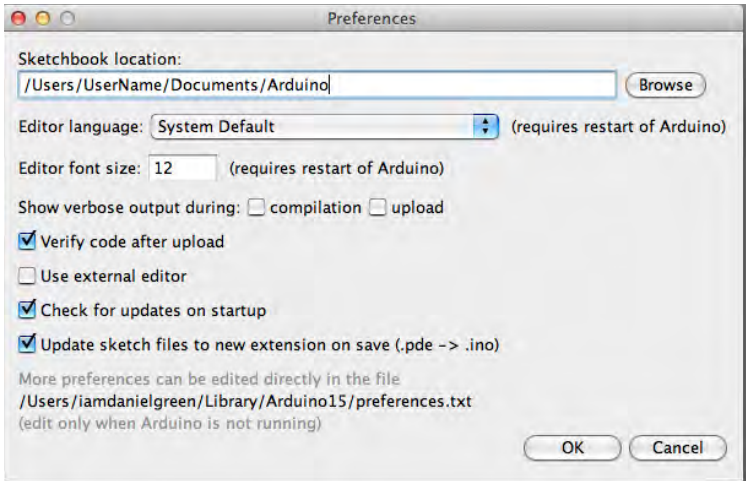
In this example, we're going to recreate the "Blink" example to test the Intel® Galileo's functionality. But first we need to install ArduBlock.

Installing ArduBlock

1. Download ardublock-all.jar from http://blog.ardublock.com/engetting-started-ardublockzhardublock/

Commencing the download will open a new window in your browser, before giving you the option to save the download. This is normal. The filename will also contain numbers, denoting the release date of the version. Still follow the instructions below.

2. In the Arduino IDE, open the menu Arduino > Preferences



3. Find "Sketchbook location"terminal.

By default, the Sketchbook location is Documents > Arduino under user's home directory.

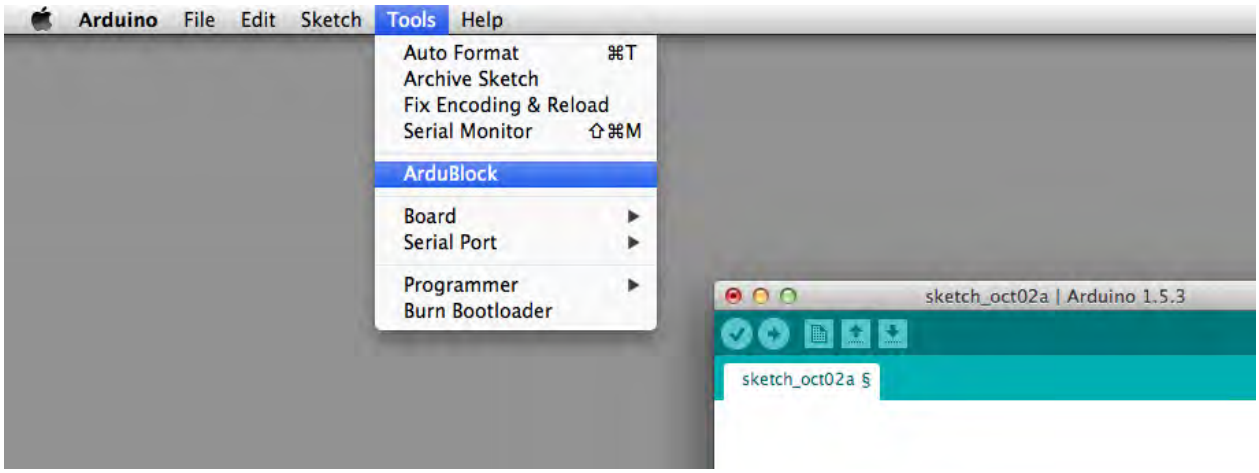
4. Copy the ardublock-all.jar file to the following directory:

Windows: C:\Users\UserName\Documents\Arduino\tools\ArduBlockTool\tool\ardublock-all.jar

Mac OSX: /Users/UserName/Documents/Arduino/tools/ArduBlockTool/tool/ardublock-all.jar

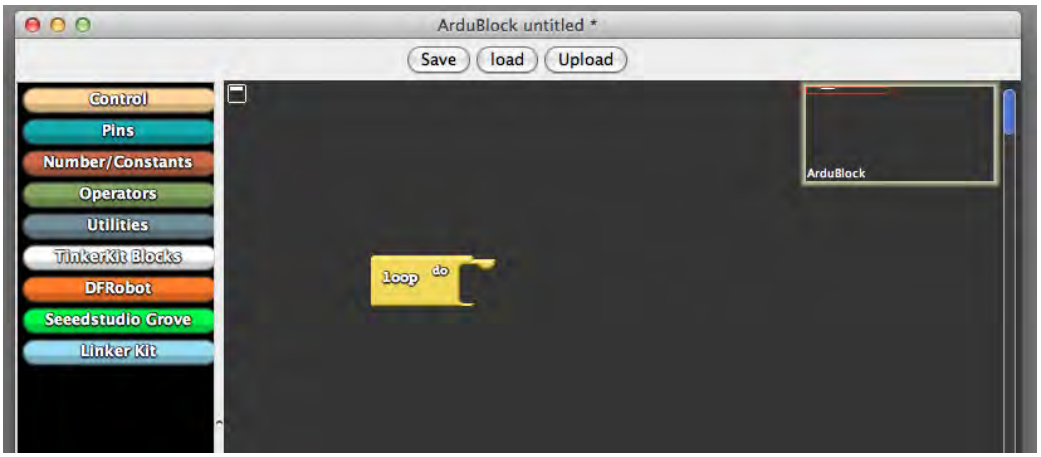
Note: ***If the above directories do not exist, these will need to be manually created. When doing so, follow the format exactly as set above, as the folder names are case sensitive.***

5. Start the Arduino IDE. You will find ArduBlock under the Tools menu, and it will open in a separate window.

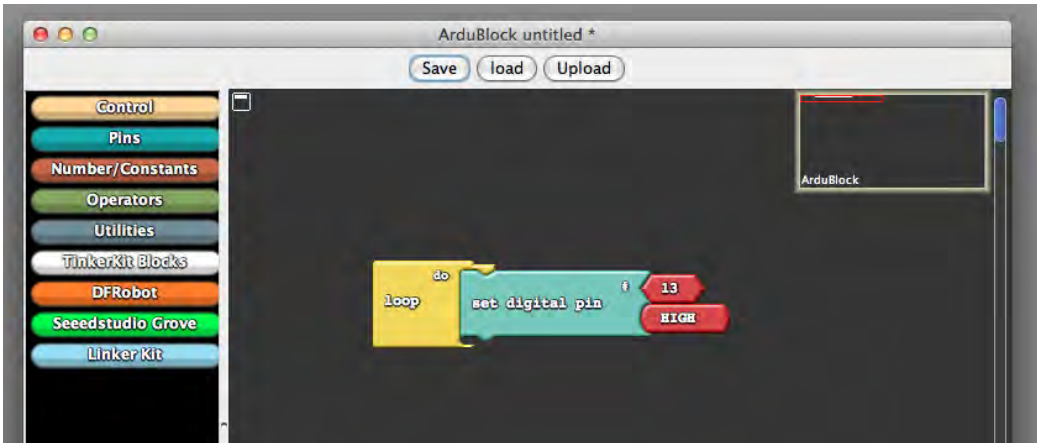


Blink with Ardublock

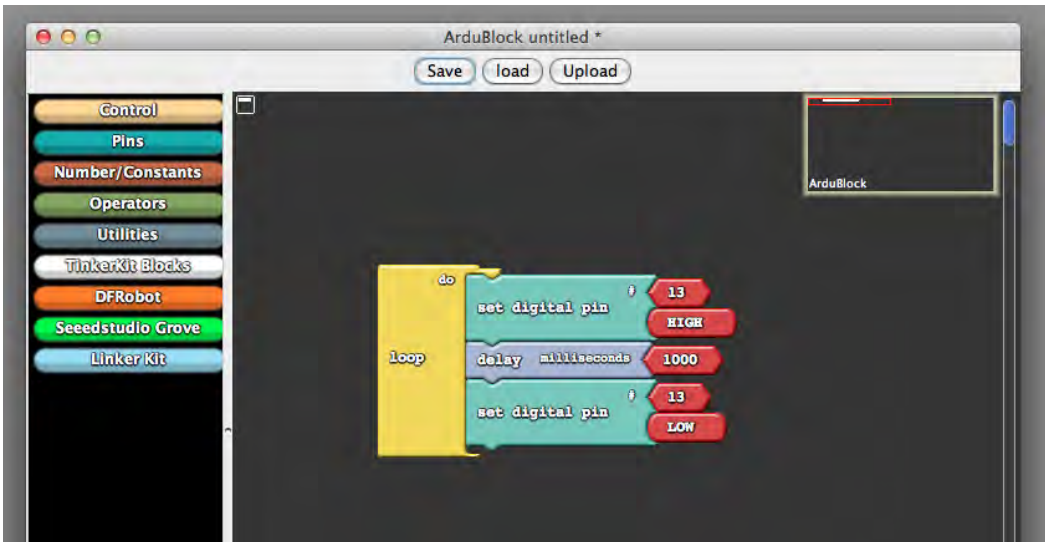
1. From the 'Control' menu, place a 'Loop' block in your work area.



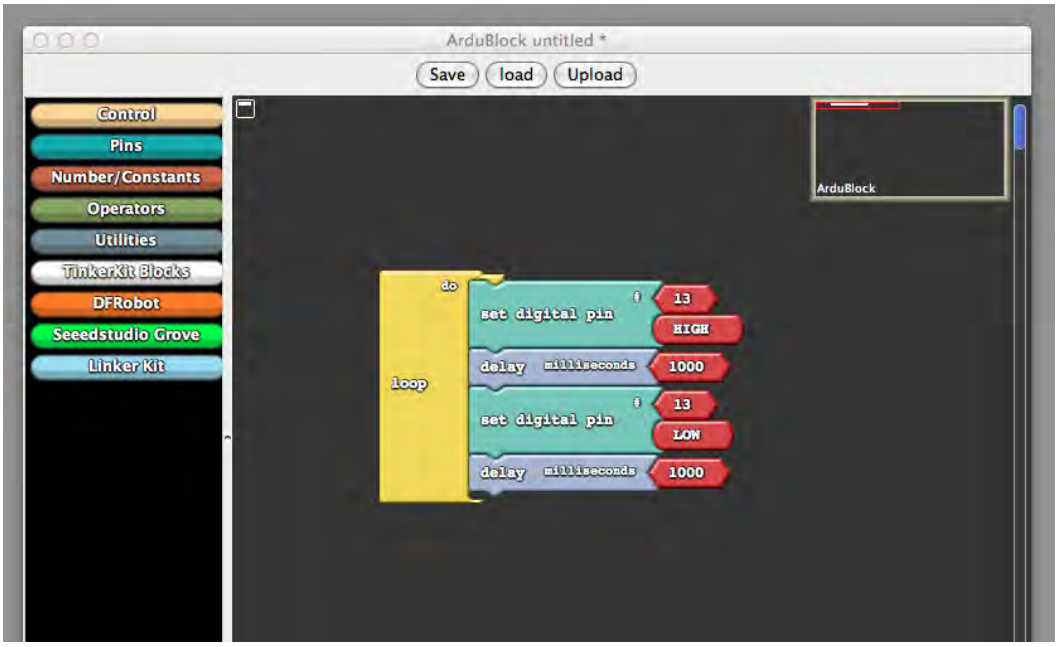
2. From the 'Pins' menu connect a 'Set Digital Pin' block to the Loop block. Set the Pin number to 13, and leave the output set to 'HIGH'. This means that when the Pin 13 of the Galileo receives power, the LED will turn on.



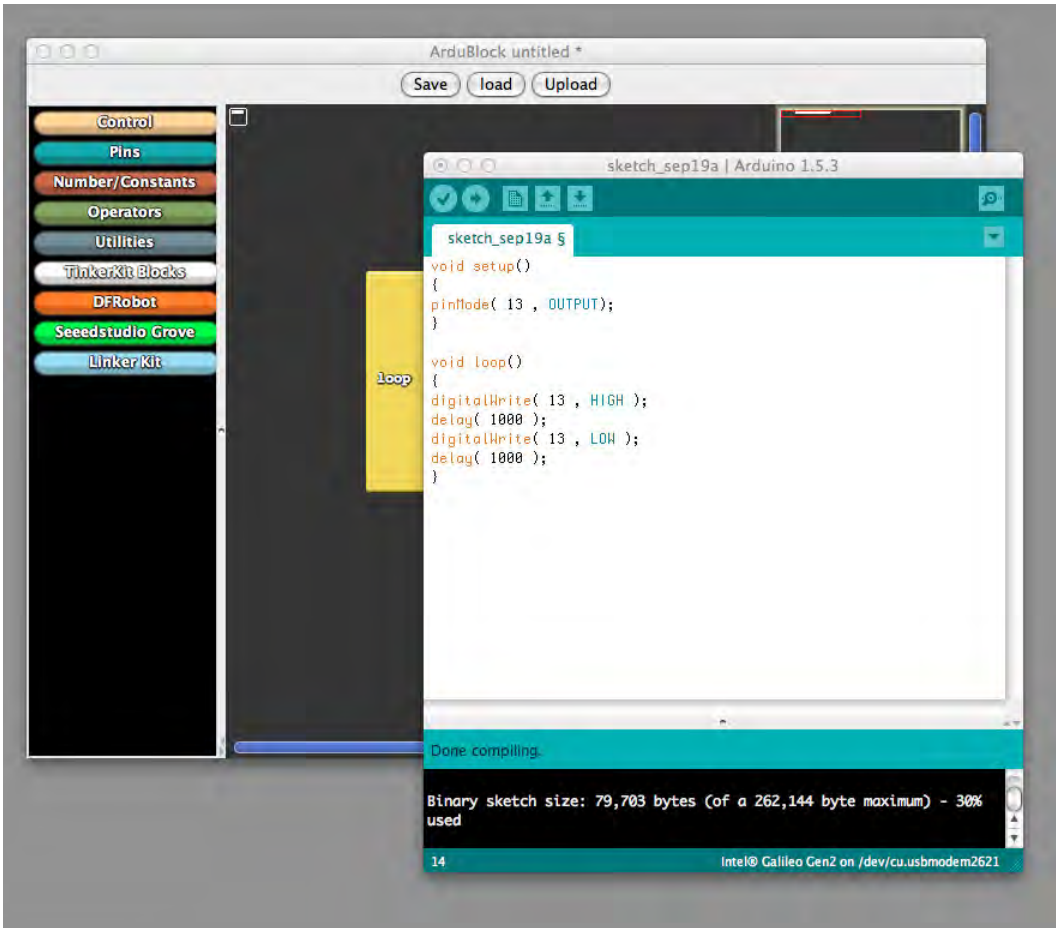
3. Connect a second 'Set Digital Pin' block to the Loop block. Set the Pin number to 13, and leave the output set to 'LOW'. This will turn the LED off again, and restart the code in the Loop block.



4. From the ‘Utilities’ menu, add a ‘Delay Milliseconds’ block underneath the two Set Digital Pin blocks. Set both delays to 1000 milliseconds. These delays will create the blinking light effect.



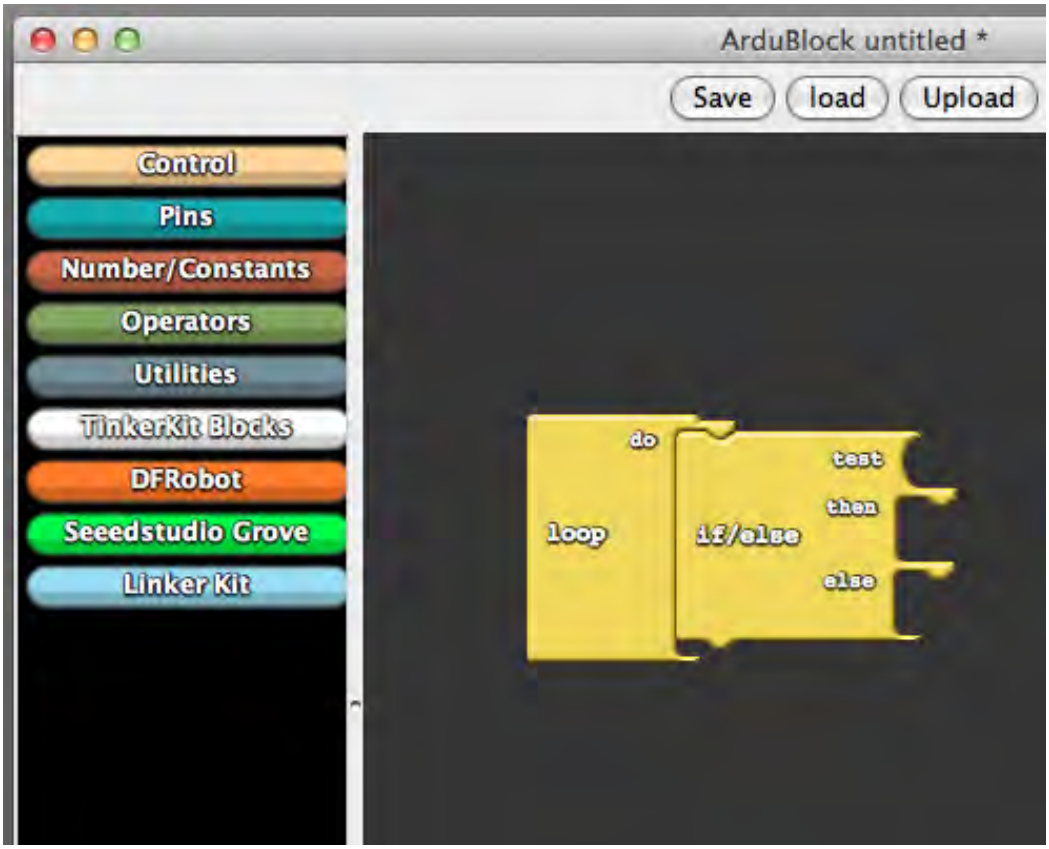
5. Click on ‘Upload’, and the text-based code will appear in the sketch window of the Arduino IDE. Click on ‘Upload’ within the Arduino IDE to send this code to the Galileo.



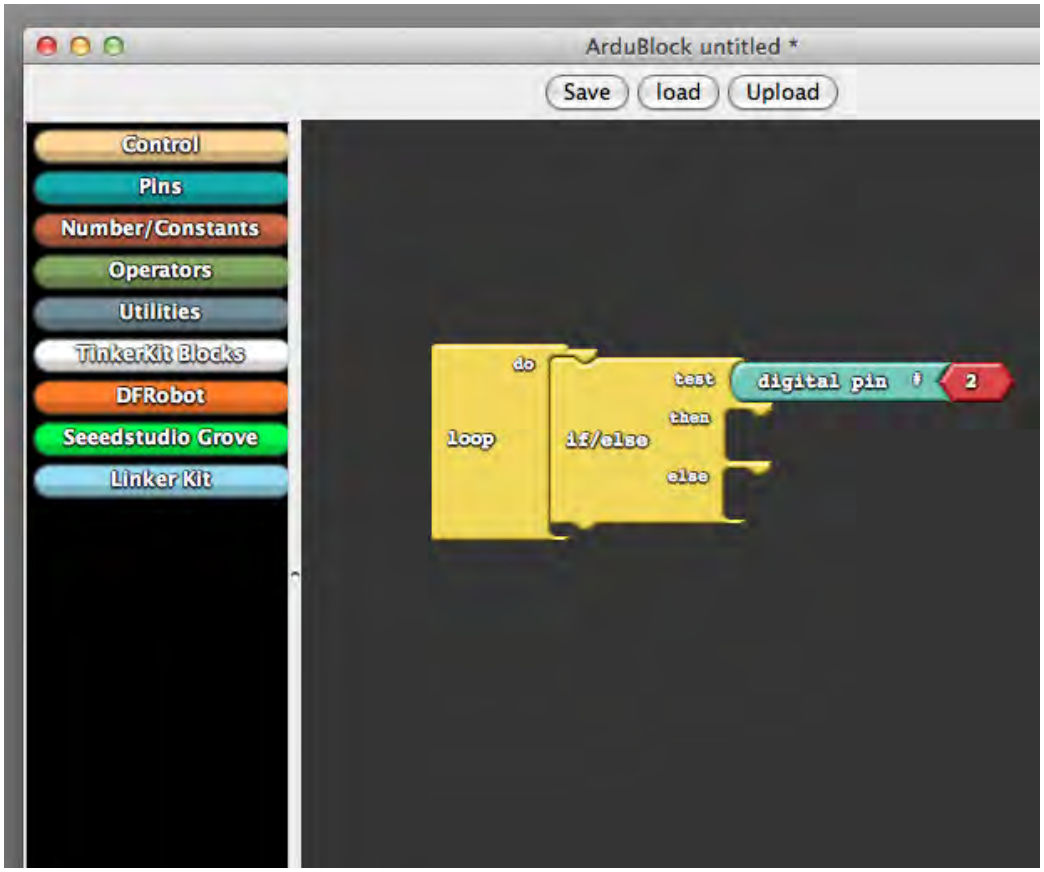
Button with Ardublock

1. From the 'Control' menu, place a 'Loop' block in your work area.

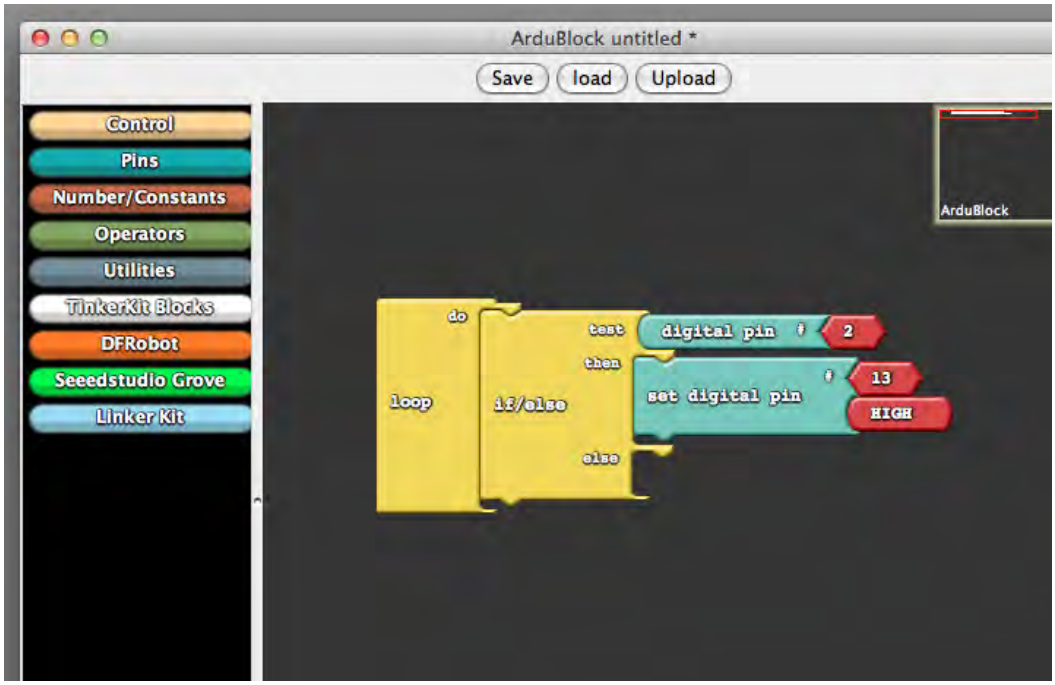
2. Then connect an 'If/Else' block to the loop block.



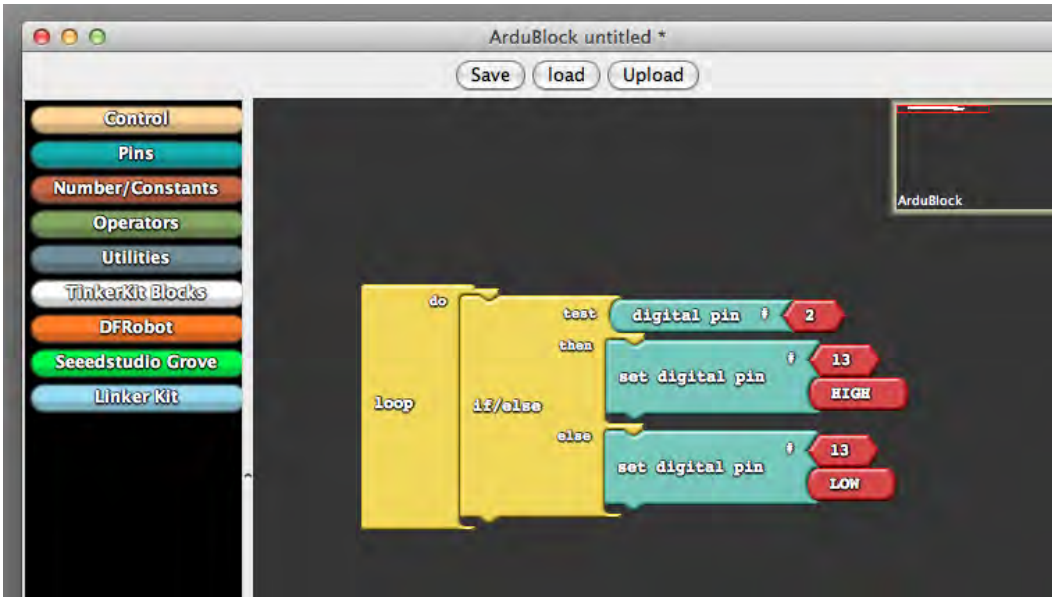
3. From the ‘Pins’ menu, connect a ‘Digital Pin #’ block to the ‘Test’ section of the If/Else block. Set the Pin number to 2. This will check for an input from Pin 2, which in this case is our button.



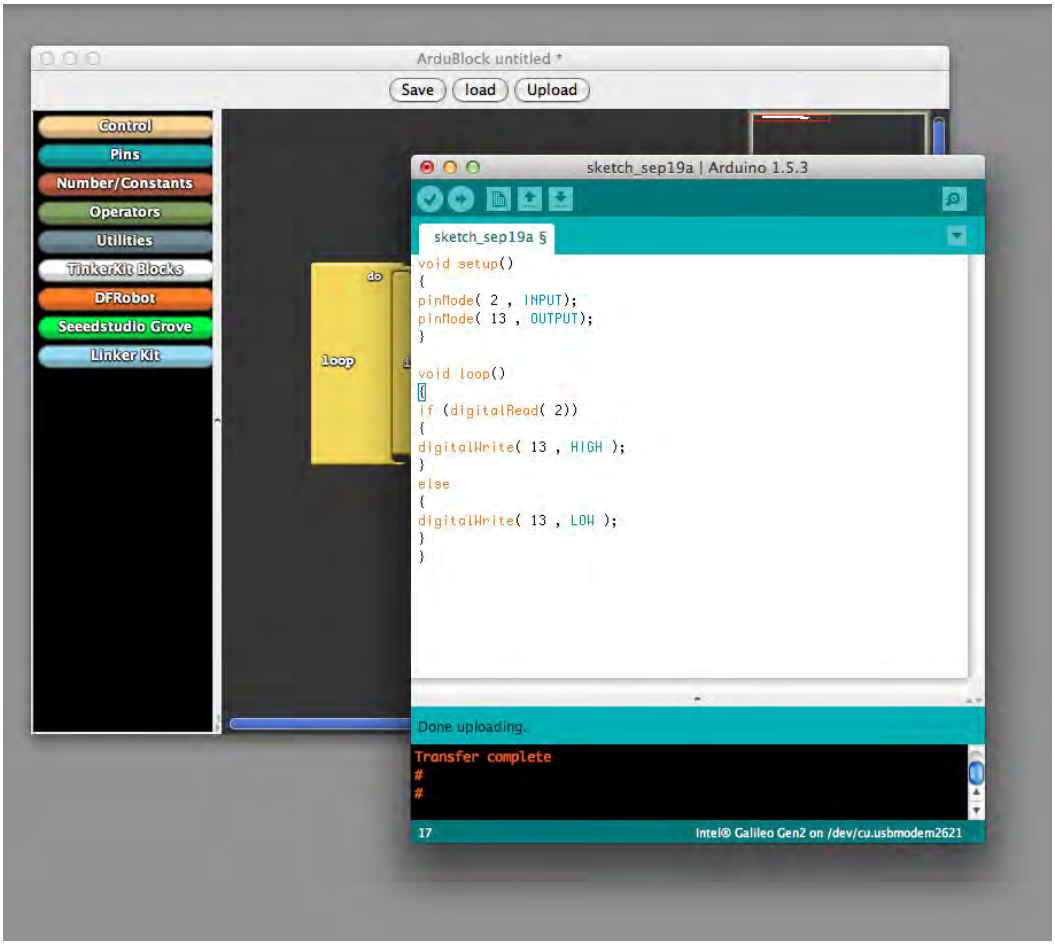
4. Then connect a ‘Set Digital Pin’ block to the ‘Then’ section of the If/Else block. Set the Pin number to 13, and leave the output set to ‘HIGH’. This means that when the button is pushed, the LED will turn on.



5. Connect a second 'Set Digital Pin' block to the 'Else' section of the If/Else block. Set the Pin number to 13, and change the output setting to 'LOW'. If the Galileo receives no input, the LED will remain off.



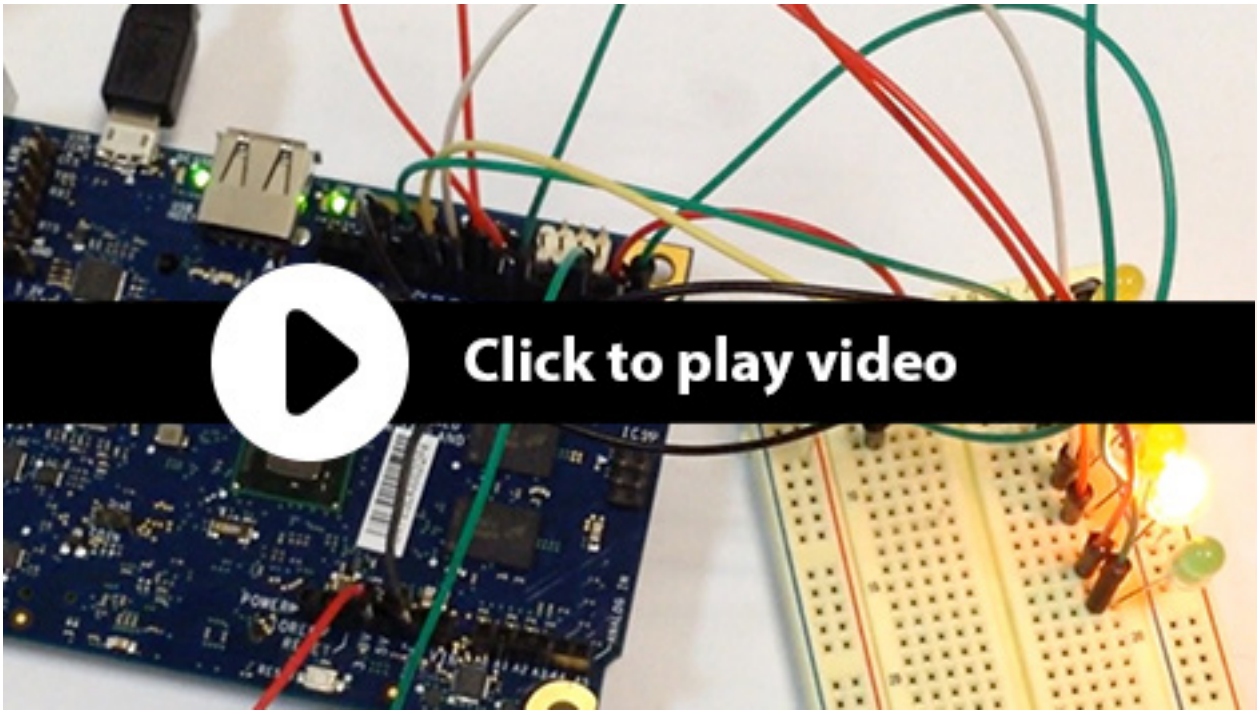
6. Click on 'Upload', and the text-based code will appear in the sketch window of the Arduino IDE. Click on 'Upload' within the Arduino IDE to send this code to the Galileo.



Building a Skill Tester: using multiple LEDs

Overview

This activity focuses on programming for a specific, and somewhat sophisticated, game functionality. A line of LEDs are programmed to illuminate in fast sequence. When a player can press a button at the same time as the last LED lights up, a green “Win” LED is triggered. If the player is too slow or fast, a red “Lose” LED is displayed. A second later the program starts again.



Materials

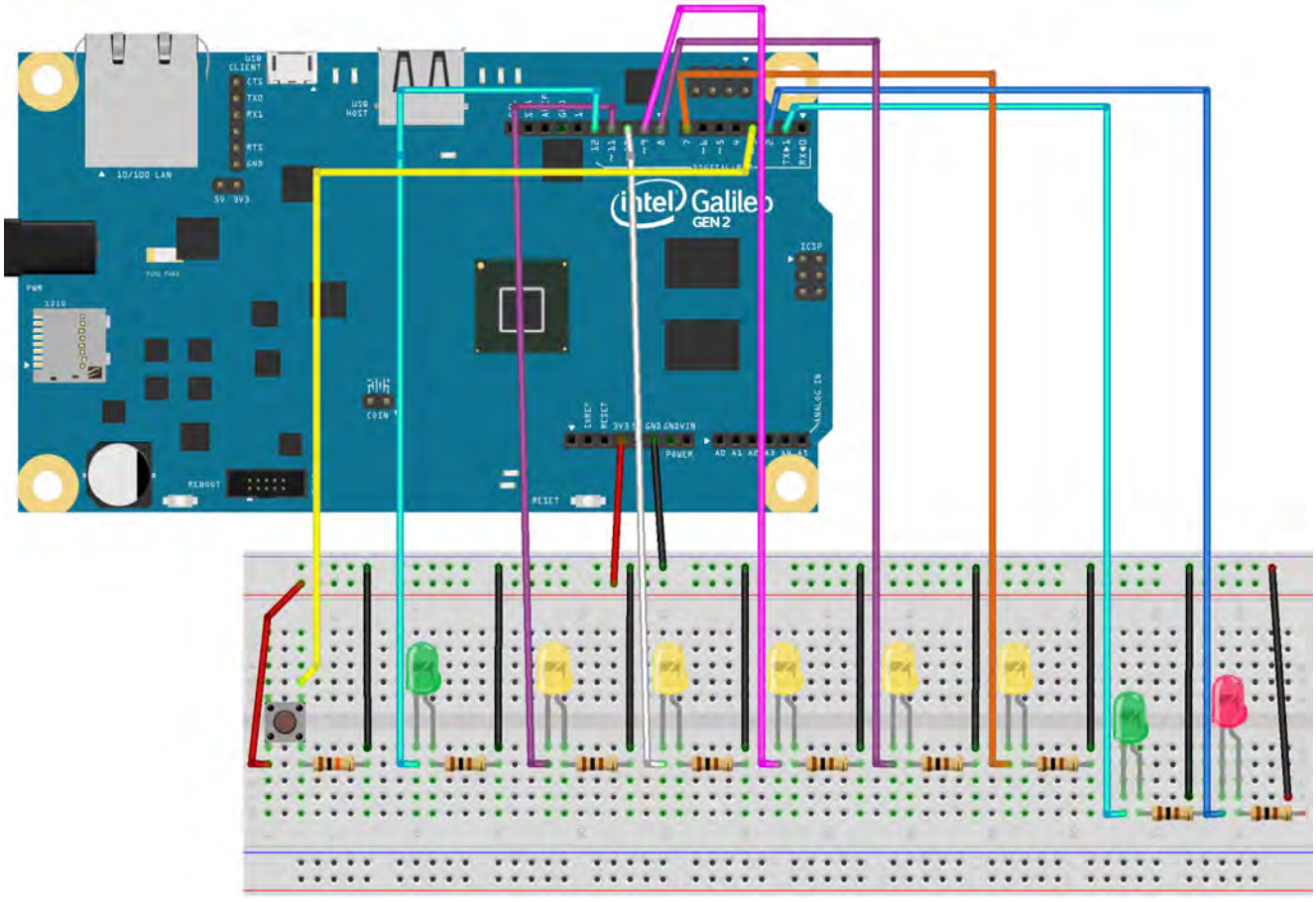
- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead
- 1 x Breadboard
- 10 x Jumper leads (M-M)
- 8 x 5mm LEDs (choose colours to suit)
- 1 x momentary “ON” push button switch
- 1 x 10K Resistor
- 8 x 100 Ω Resistors

Wiring

1. Wire up the LEDs and 100 Ω pull-down resistors as per the diagram. To do this, connect the negative (shorter) lead of the LED to one end of the 100 Ω resistor. Connect the other end of the resistor to the ground rail.

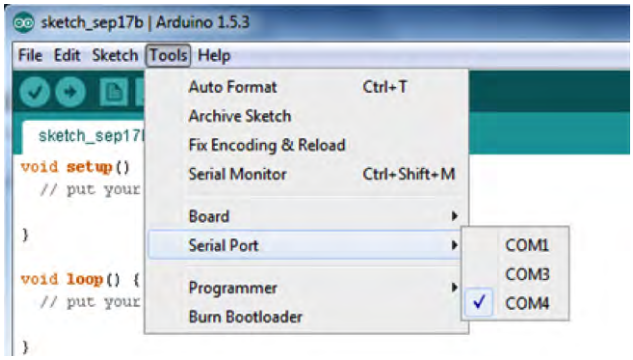
Note: ***Remember breadboards are connected vertically so the LED and the resistor should be in the same vertical line.***

2. Connect the longer positive leads of the Game light LEDs (6 LEDs in a row) to pins 7 to 12 with the green LED connecting to Pin 7 and the others consecutively to Pins 8 – 12.
3. Connect the longer positive leads of the Result LEDS (green and red) to Pins 1 and 2 respectively.
4. Place push button on the breadboard. Connect one end of the 10K Ω resistor to one of the pusbutton pins. Connect the other end of the resistor to the ground rail. Connect the ground rails on the breadboard together using a jumper lead.
5. Connect a wire from the other leg on the same side as the resistor to the 3.3V pin.
6. Finally, connect a wire from one of the legs on the opposite side of the pushbutton to Pin 3.

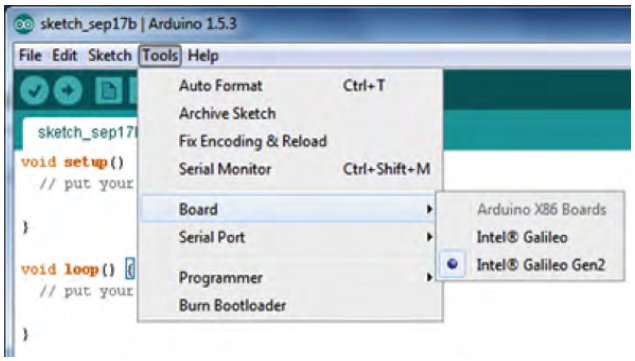


Coding

Open the Arduino IDE. Check that the COM port assigned to the Intel® board (go to Device Manager to check), matches the correct COM port according the the IDE:



Check also that the correct Intel® board has been selected:



Copy and paste the code below. (You can alter the level of difficulty by changing the speed in the LED daisy chain coding.)

```
/*
  Reaction Game
  Turns on a series of 5 yellow LEDs in sequence.
  There is a 6th green LED and the player must touch the switch when the
  green LED lights up.

  This example code is in the public domain.
 */

// Assign variables to the inputs/outputs
int greenLED = 1;
int redLED = 2;

int ledPin12 = 12;
int ledPin11 = 11;
int ledPin10 = 10;
int ledPin9 = 9;
int ledPin8 = 8;
int ledPin7 = 7;
```

```
int switchPin = 3;
int switchPinState = 0;
int lastSwitchPinState = 0;
int switchPinStateBeforeLightingLed7 = 0;

int delayBetweenLEDs = 100;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pins corresponding to the LEDs as outputs.
  pinMode(ledPin12, OUTPUT);
  pinMode(ledPin11, OUTPUT);
  pinMode(ledPin10, OUTPUT);
  pinMode(ledPin9, OUTPUT);
  pinMode(ledPin8, OUTPUT);
  pinMode(ledPin7, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(greenLED, OUTPUT);

  // initialise the switch digital pin as an input
  pinMode(switchPin, INPUT);

}

// the loop routine runs over and over again forever:
void loop() {

  // start the LEDs off
  digitalWrite(redLED,LOW);
  digitalWrite(greenLED,LOW);

  digitalWrite(ledPin12,HIGH);
  delay(delayBetweenLEDs);
  digitalWrite(ledPin12,LOW);
  digitalWrite(ledPin11,HIGH);
  delay(delayBetweenLEDs);
  digitalWrite(ledPin11,LOW);
  digitalWrite(ledPin10,HIGH);
  delay(delayBetweenLEDs);
  digitalWrite(ledPin10,LOW);
  digitalWrite(ledPin9,HIGH);
  delay(delayBetweenLEDs);
  digitalWrite(ledPin9,LOW);
  digitalWrite(ledPin8,HIGH);
  delay(delayBetweenLEDs);
  digitalWrite(ledPin8,LOW);
  switchPinStateBeforeLightingLed7 = digitalRead(switchPin);
  digitalWrite(ledPin7,HIGH);
```

```
  for (int i=0; i < 10000; i++) {
    if (!switchPinStateBeforeLightingLed7) { // not allowed to hold down button!
          switchPinState = digitalRead(switchPin);
          if (switchPinState == HIGH) {
            digitalWrite(greenLED,HIGH);
            digitalWrite(redLED,LOW);
            break;
          }
          else {
              digitalWrite(redLED, HIGH);
               digitalWrite(greenLED, LOW);
          }
    }
    else {
      digitalWrite(redLED, HIGH);
    }
  }
  delay(250); // wait to let person see how they went
  digitalWrite(ledPin7,LOW);
}
```

Download and run program.

Explanation of the code

The first part of the code assigns variables to the input and output pins to make the code more readable. For example, Pin 1 is assigned the variable, “greenLED” in the following line:

```
int greenLED = 1;
```

Some other variables are set up to monitor if the button is pressed too early to prevent people from cheating:

```
int switchPinState = 0;
int lastSwitchPinState = 0;
int switchPinStateBeforeLightingLed7 = 0;
```

There is also a variable set up to represent the flashing delay:

```
int delayBetweenLEDs = 100;
```

In the setup() procedure, the pins are initialised to be either INPUT (for the pushbutton) or OUTPUT, for the LEDS:

```
  // initialize the digital pins corresponding to the LEDs as outputs.
  pinMode(ledPin12, OUTPUT);
  pinMode(ledPin11, OUTPUT);
  pinMode(ledPin10, OUTPUT);
  pinMode(ledPin9, OUTPUT);
  pinMode(ledPin8, OUTPUT);
  pinMode(ledPin7, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
```

```
// initialise the switch digital pin as an input
pinMode(switchPin, INPUT);
```

In the loop, the main logic occurs.

First, the red and green result LEDs are set to LOW:

```
digitalWrite(redLED,LOW);
digitalWrite(greenLED,LOW);
```

Then six LEDs are sequenced to flash on (HIGH) and off(LOW). For example, the following three lines set the LED connected to pin 12 to flash on and off:

```
digitalWrite(ledPin12,HIGH);
delay(delayBetweenLEDs);
digitalWrite(ledPin12,LOW);
```

Before writing Pin 7 HIGH, the state of the push button is read:

```
switchPinStateBeforeLightingLed7 = digitalRead(switchPin);
```

The next part of the code then checks for the button to be pressed while the green LED is lit. It uses a for loop to check 10000 times for the button being pressed. The following line starts the for loop and the code inside the starting and ending brace { is performed 10000 times.

```
for (int i=0; i < 10000; i++) {
```

If it finds the button is pressed it sets the green result LED to HIGH and sets the red result LED to LOW and calls the break command to jump out of the for loop. If the button is not pressed it sets the red result LED to HIGH.

```
for (int i=0; i < 10000; i++) {
  if (!switchPinStateBeforeLightingLed7) { // not allowed to hold down button!
        switchPinState = digitalRead(switchPin);
        if (switchPinState == HIGH) {
          digitalWrite(greenLED,HIGH);
          digitalWrite(redLED,LOW);
          break;
        }
        else {
            digitalWrite(redLED, HIGH);
             digitalWrite(greenLED, LOW);
        }
  }
  else {
    digitalWrite(redLED, HIGH);
  }
}
```

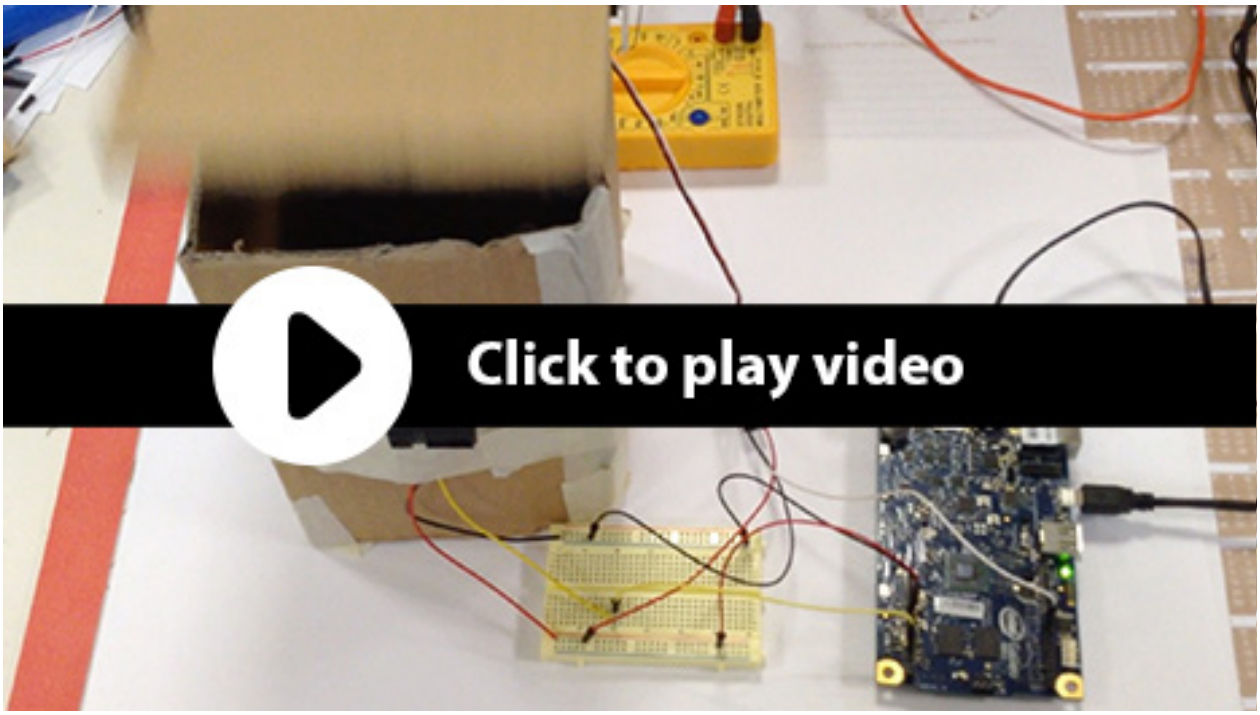
Finally, a delay is called to let the person see how they went and then the Green LED is set to LOW:

```
delay(250); // wait to let person see how they went
digitalWrite(ledPin7,LOW);
```

Building an Automatic bin: using infra-red motion sensor and micro servo

Overview

This project combines the use of a proximity infra-red motion sensor as an input to the Intel® Galileo Gen 2 board with a micro servo as an output, to operate a lid to a cardboard desktop bin – handy for all those little rubbish items you want covered and out of sight. Programming code is minimal and can be easily modified to adjust the trigger distance and servo swing (on and off operation). Some basic building and manipulation required.



Materials

- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead
- 1 x Breadboard
- 6 x Jumper leads (M-M)
- 1 x Infrared motion sensor with 3pin plug and lead
  (for this test we used Infrared Proximity Sensor - Sharp GP2Y0A21YK and 3pin JST connector lead)
- 1 x Mini servo
  (for this test we used Arduino T010050 Micro servo)

OTHER ITEMS REQUIRED

- 1 x steel paper clip
- Masking tape
- Re-used cardboard box (or off-cut)

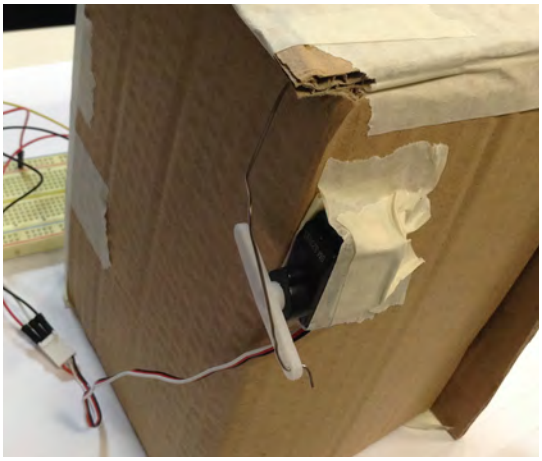
Hardware construction

1. Create a small cardboard bin using parts of a larger cardboard box, and bending to suit.

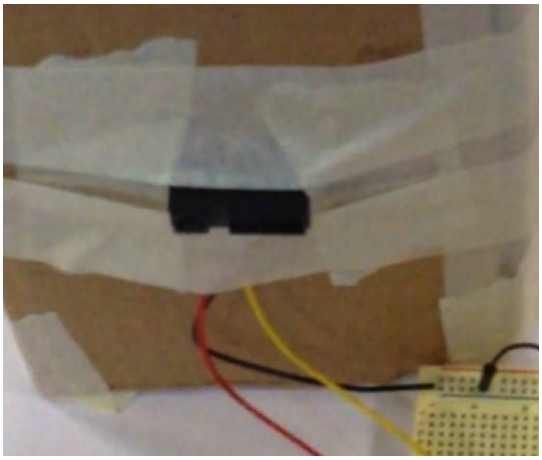
2. Masking tape can be used for holding parts together. Alternatively, practise joining cardboard pieces using a wedge & groove method.

3. Cut out cardboard pieces for base and lid of bin and attach with masking tape. Masking tape can also be used as the hinge for the bin lid.

4. Fit micro servo to back of box using masking tape and modify paper clip with pointy nose pliers to act as push rod between servo arm and bin lid.



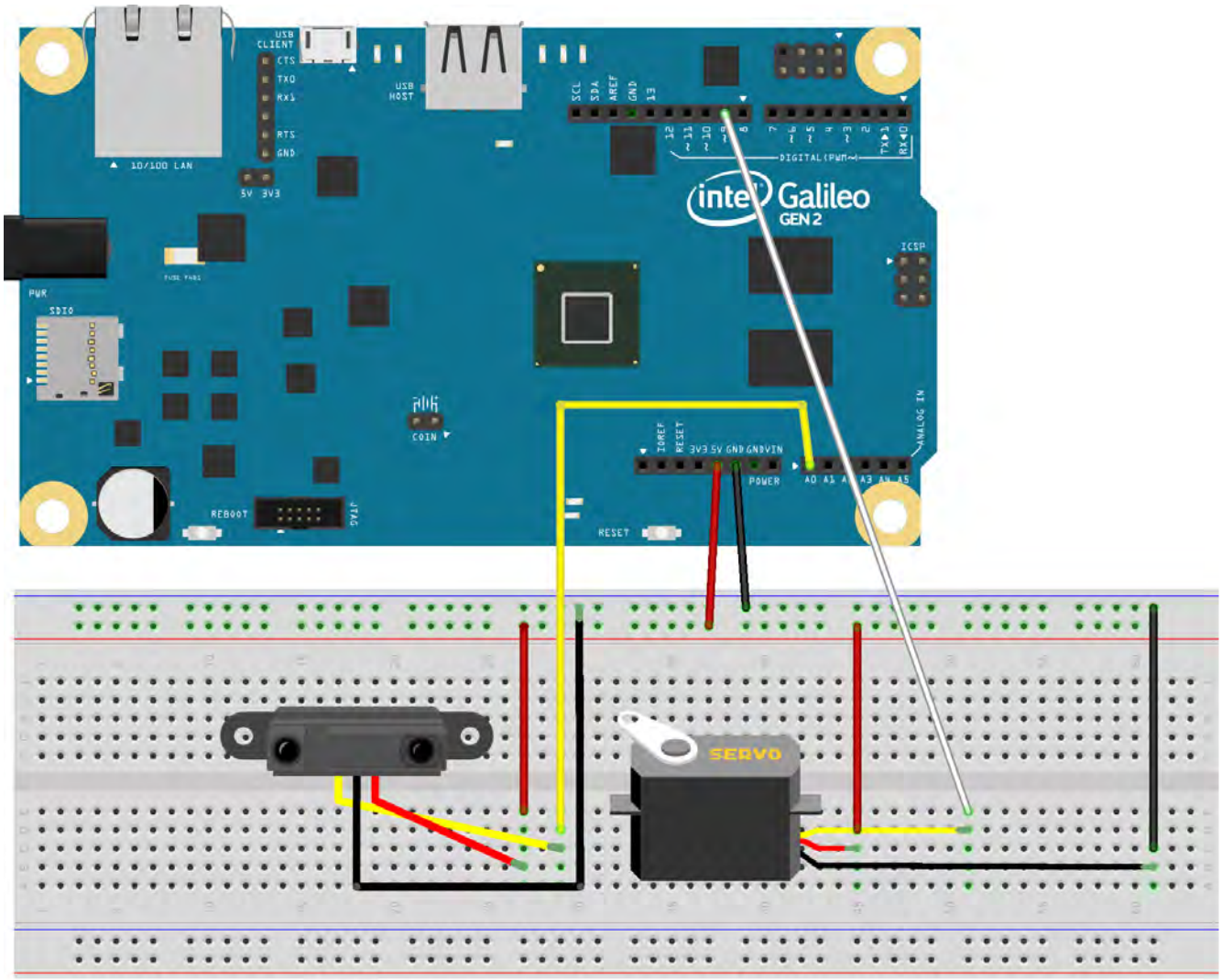
5. Fit infra-red sensor to front of bin using masking tape.



Optional: You can create a removable base to conceal breadboard and Intel® Galileo Gen 2 board. 3pin JST lead from infra-red sensor can be inserted through front of bin by making a small hole through the cardboard.

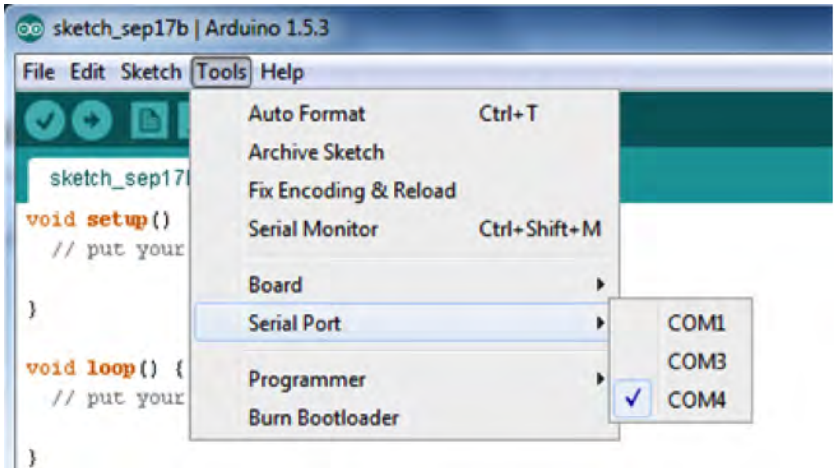
Wiring

1. Using M-M jumper wires, connect common ground and positive wires from Infra-red sensor and servo to breadboard.
2. Connect analogue input wire (yellow) from infra-red sensor to breadboard using a M-M jumper wire to pin A0 on Intel® Galileo Gen 2 board.
3. Connect digital input wire from micro servo (white) to pin 9 on Intel® Galileo Gen 2 board.
4. Finally, connect ground and 5V supply using M-M jumper wires from breadboard to pins GND and 5V of Intel® Galileo gen 2 board, respectively.

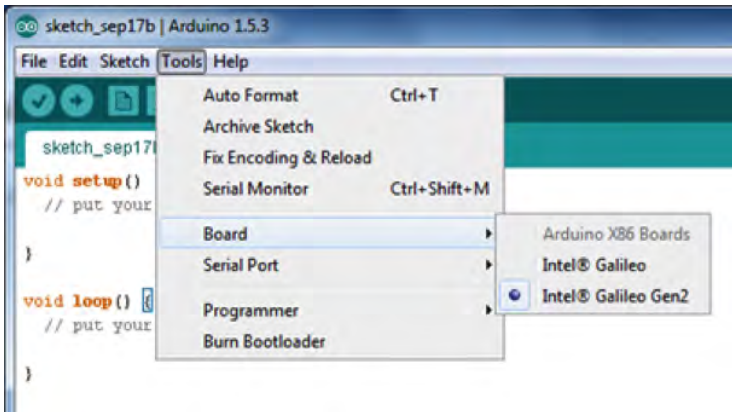


Coding

1. Open the Arduino IDE. Check that the COM port assigned to the Intel® board (go to Device Manager to check), matches the correct COM port according the the IDE.



2. Check also that the correct Intel® board has been selected.



3. Copy and paste the code below, noting values set for both infra-red sensor and micro servo. (Example will trigger servo within 70cm from object. You can change values to suit requirement.)

```
#include <Servo.h>  // this provides code for the Servo
Servo myServo;  // create servo object to control a servo
int infraredSensor = 0;  // analog pin used to connect the potentiometer
int sensorValue;   // variable to read the value from the analog pin
void setup()
{
  myServo.attach(9);  // attaches the servo on pin 9 to the servo object
  Serial.println("In setup ");
}
void loop()
{
 sensorValue = analogRead(infraredSensor);   // reads the value of the infrared sensor
(value between 0 and 1023)
  Serial.println("Motion sensor value is ");
  Serial.print(sensorValue); // write some debugs to the Serial console
  if (sensorValue > 100) {
  myServo.write(179);   // write the maximum value of the Servo, 179
  }
  delay(1000); // hold the bin lid up for one second
  myServo.write(0); // push the bin lid back down
  }
```

Download and run program.

Explanation of the Code

The first line of the program includes some prewritten code that is specific to servo motors on the Arduino:

```
#include <Servo.h>  // this provides code for the Servo
```

The following lines of code initialise variables to represent the servo motor, the infrared sensor and the value returned from the sensor:

```
Servo myServo;  // create servo object to control a servo
int infraredSensor = 0;  // analog pin used to connect the potentiometer
int sensorValue;   // variable to read the value from the analog pin
```

In the setup code, the first line sets up a servo to be attached to Pin 9 while the second line is a debug statement to print to the serial console:

```
myServo.attach(9);  // attaches the servo on pin 9 to the servo object
Serial.println("In setup ");
```

In the loop code, the first line reads the value from the infrared sensor followed by a couple of debug statements to print out the sensor value to the serial console:

```
sensorValue = analogRead(infraredSensor);   // reads the value of the infrared sensor
(value between 0 and 1023)
Serial.println("Motion sensor value is ");
Serial.print(sensorValue); // write some debugs to the Serial console
```

The following lines check that the value of the sensor is greater than 100 and writes a maximum value to the servo to make it move the bin lid up:

```
if (sensorValue > 100) {
  myServo.write(179);   // write the maximum value of the Servo, 179
}
```

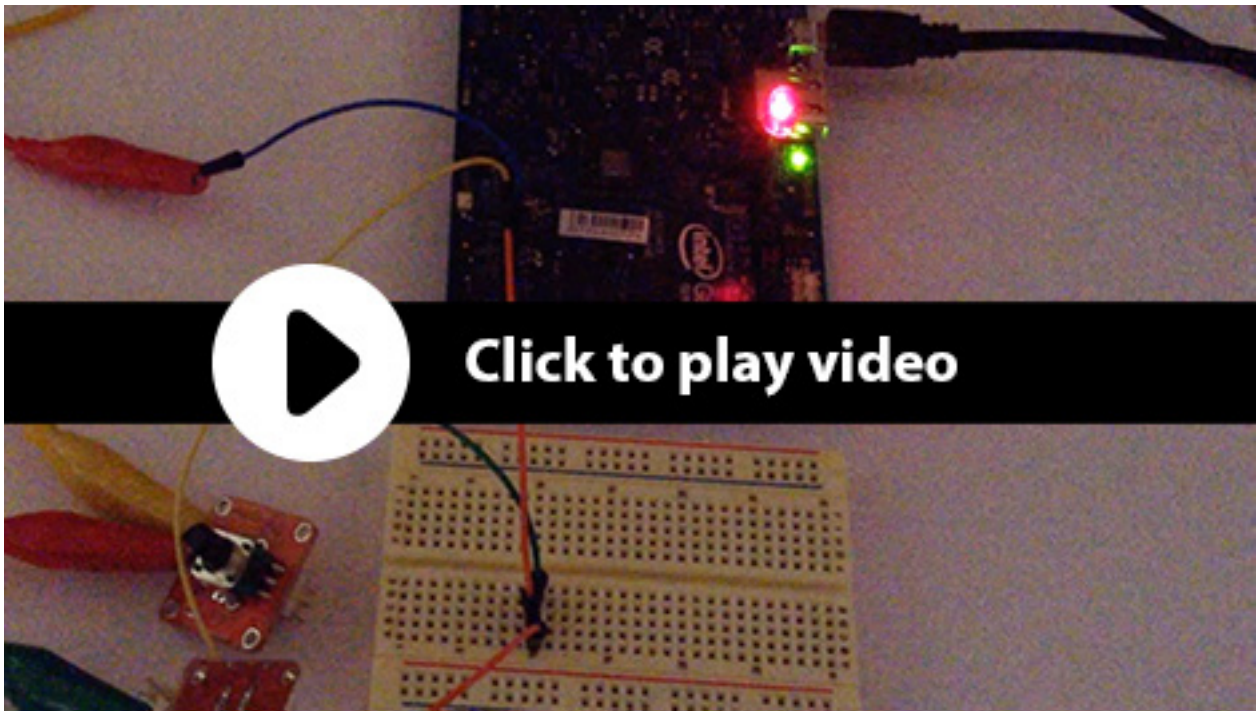
Then a delay is added to hold the bin lid up and 0 is written to the servo to push the bin lid back down:

```
delay(1000); // hold the bin lid up for one second
myServo.write(0); // push the bin lid back down
```

Building a night light: using a photocell and LED

Overview

This activity combines the use of a photocell (light-dependent resistor) as an input to the Intel® Galileo Gen 2 board with an LED as an output, to operate when in dark and turn off when in light (a night light). Programming code is minimal and operation can easily be calibrated, using a 10K potentiometer, to any room of the house. Some basic paper cutting and manipulation is required to enhance the lighting effect.

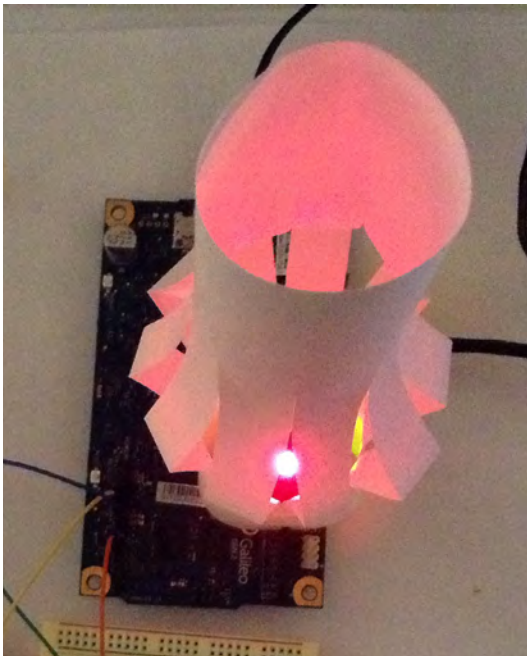


Materials

- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead
- 1 x Breadboard
- 4 x Jumper leads (M-F)
- 1 x Jumper lead (M-M)
- 1 x Photocell
- 1 x 5mm red LED
- 100 Ω Resistor

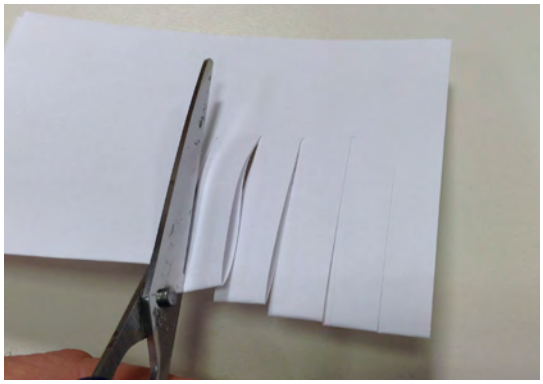
OTHER ITEMS REQUIRED

- 1 x A4 sheet of white paper
- Masking tape

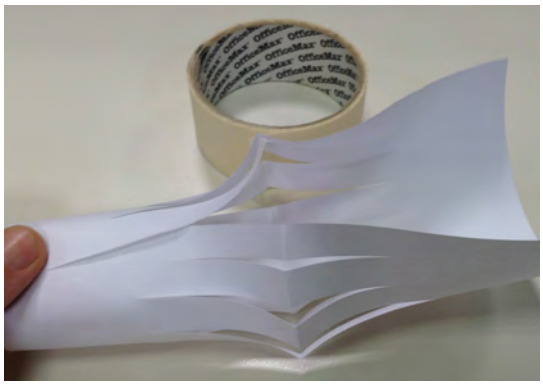


Hardware construction

1. To make the night light diffuser fold an A4 piece of paper into A5 size, fold again and cut strips along the fold.

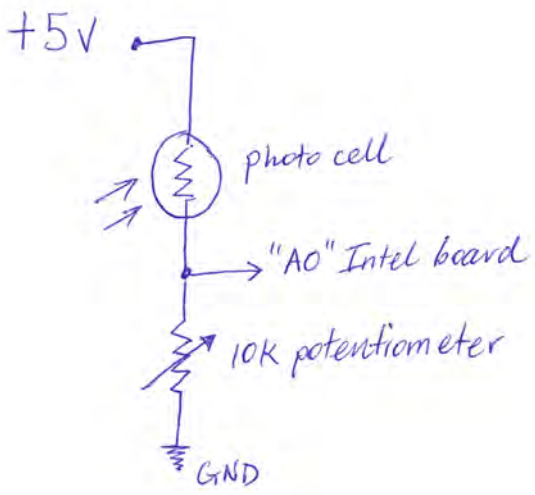


2. Roll paper long ways and tape into a tubular shape and use masking tape to hold ends together.



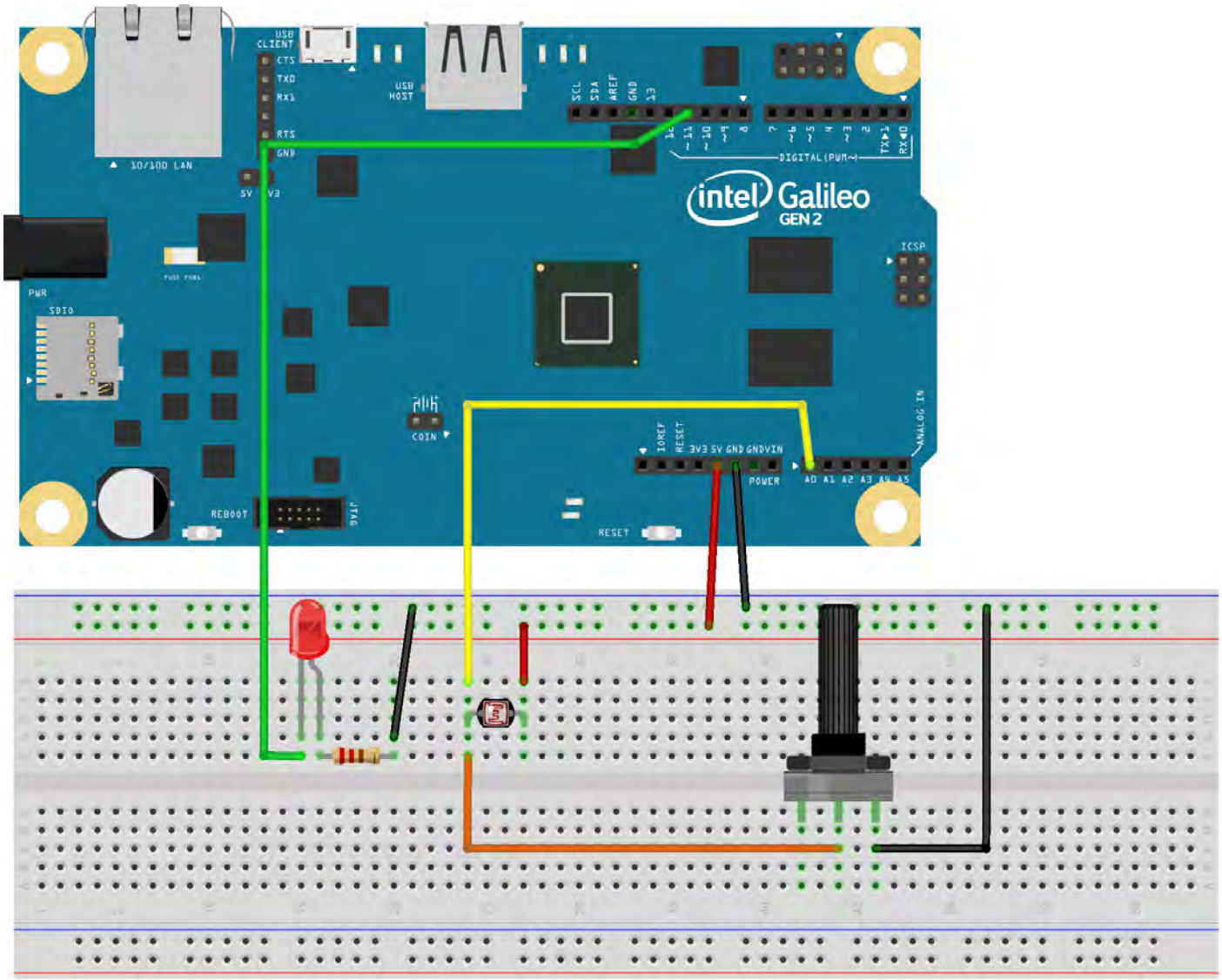
Wiring

Refer to circuit diagram for wiring of individual components:



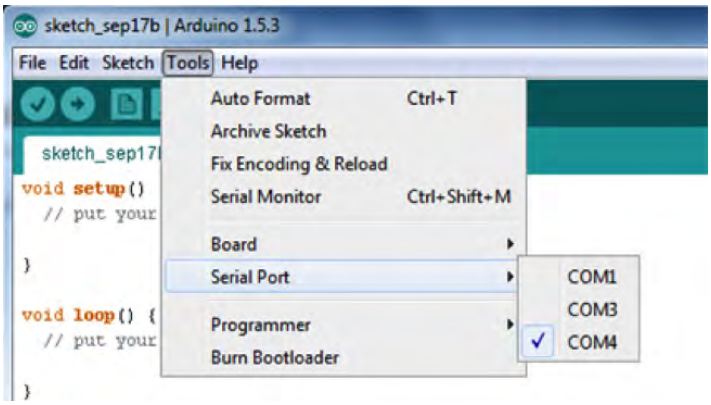
1. Connect positive lead of LED Pin 11 and twist a resistor around the negative lead, connecting the other end of the resistor to GND.

2. Connect one end of photocell to 5V on the board. Connect the other end of the photocell to the ‘middle’ leg of the potentiometer. Also connect a wire from the other end of the photocell to A0 on the board.
3. Connect one of the outer legs on the potentiometer to GND on the board.
4. Connect part of the photocell and potentiometer, as indicated in the circuit below, onto the breadboard, using M-F jumper leads.
5. Connect breadboard to Intel® Galileo Gen 2 board using selection of M-M and M-F jumper leads.
6. Connect LED to pins 11 (long lead) and GND (short lead).
7. Place paper diffuser onto LED.

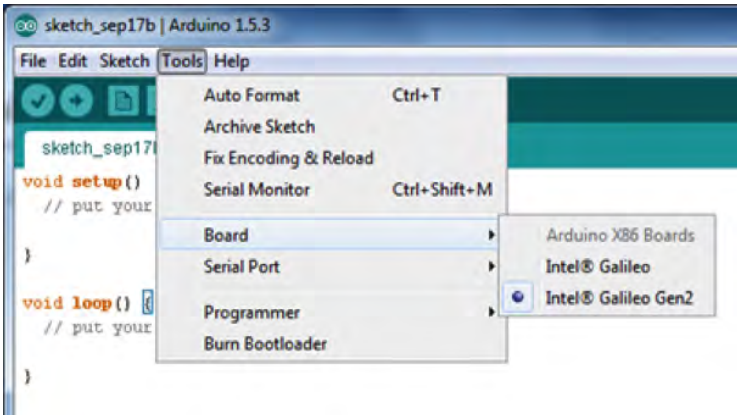


Coding

1. Open the Arduino IDE. Check that the COM port assigned to the Intel® board (go to 'Device Manager' to check), matches the correct COM port according the the IDE.



2. Check also that the correct Intel® board has been selected.



3. Copy and paste the code below, noting value set for photocell. (You can calibrate to your room setting using the 10K potentiometer.)

```
int ledPin = 11;    // LED connected to digital pin 11
const int analogInPin = 0;  // Analog input pin that the potentiometer is attached to
int sensorValue = 0;        // value read from the pot
int outputValue = 0;
void setup()  {
  pinMode(ledPin, OUTPUT); //
  Serial.begin(9600);
}
void loop()  {
   // read the analog in value:
   sensorValue = analogRead(analogInPin);
   Serial.print("sensor value in is ");
   Serial.println(sensorValue);
   // logic for night light
   outputValue = map(sensorValue, 0, 1023, 0, 255);
   if (outputValue < 102) {
     analogWrite(ledPin, outputValue);
```

```
    }
    else {
      analogWrite(ledPin, 0);
      Serial.println('Writing 0 as it is > 40'); // daytime so LED is off
    }
    delay(1000);
}
```

Download and run program.

Explanation of the Code

The first four lines of the program initialise some variables to represent the LED pin, the analog pin, the sensor value read from the potentiometer and the output value that will be written to the LED:

```
int ledPin = 11;    // LED connected to digital pin 11
const int analogInPin = 0;  // Analog input pin that the potentiometer is attached to
int sensorValue = 0;        // value read from the pot
int outputValue = 0;
```

In the setup code, the LED pin is set to be of OUTPUT type and the serial console is initialised. You must have 9600 for the baud rate.

```
pinMode(ledPin, OUTPUT); //
Serial.begin(9600);
```

In the loop code, the value is read from the analog pin and stored in the sensorValue variable followed by some debug statements to the serial console printing what the value is:

```
sensorValue = analogRead(analogInPin);
Serial.print("sensor value in is ");
Serial.println(sensorValue);
```

Using the built in map function, the value from the analog pin is mapped from the range of [0, 1023] to the LED range of [0, 255] with the following command:

```
outputValue = map(sensorValue, 0, 1023, 0, 255);
```

The outputValue returned from the previous command is then checked to see if it is less than a certain threshold. In the case of darkness, the photocell will provide a large resistance and lower the voltage returned by the Analog pin, AO. When the outputValue is lower than the threshold, the LED will be turned on by writing the outputValue to the LED pin. Otherwise, a 0 value is written to the LED. Finally a delay is programmed to make sure the light stays on momentarily:

```
if (outputValue < 102) {
  analogWrite(ledPin, outputValue);
}
else {
  analogWrite(ledPin, 0);
  Serial.println('Writing 0 as it is > 40'); // daytime so LED is off
}
delay(1000);
```

USING THE GALILEO AS A COMPUTER

Communicating with the Galileo

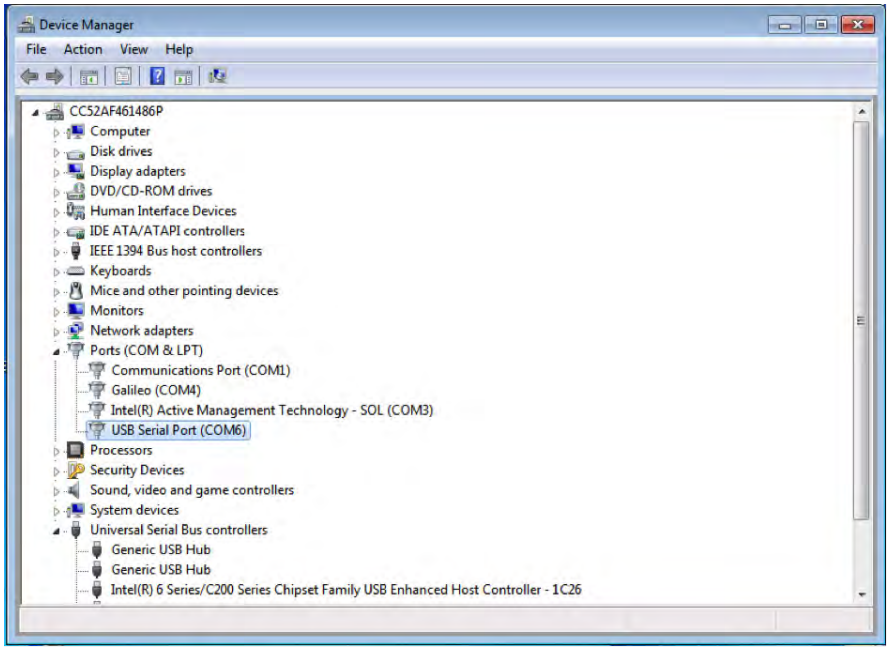
Connecting via Serial

You can communicate serially with your Intel® Galileo Gen 2 via the 6-pin FTDI serial connector and the USB port on your computer.

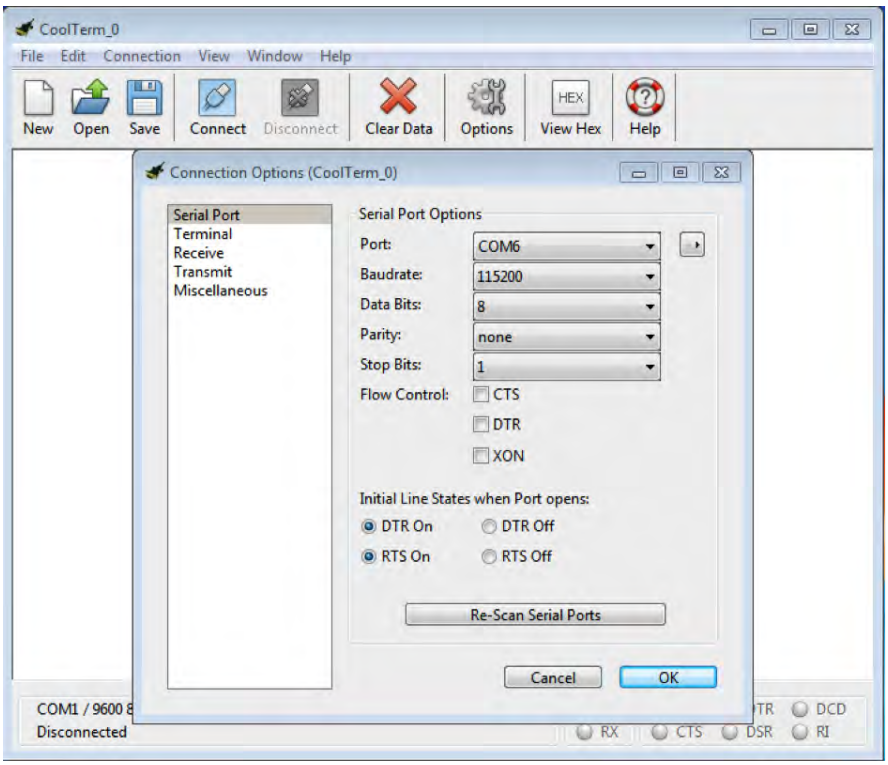
This is also known as a USB to TTL Serial Cable (3.3V). The connector slips right on, and the Intel® Galileo Gen 2 board has the identifiers GRN and BLK silkscreened onto the surface, to help you align the cable with the wire colors on the FTDI connector hood. The 6-pin header operates at 3.3 V TTL levels.

You can run Linux commands serially using this cable and some serial communication software such as the free CoolTerm (http://freeware.the-meiers.org).

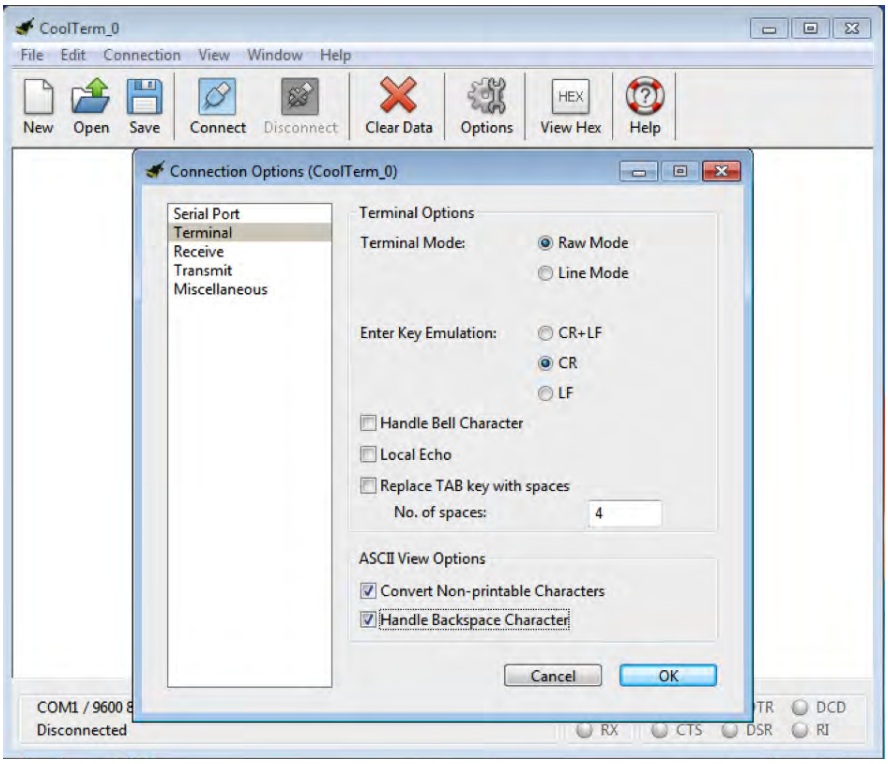
1. Connect your Serial Cable to your computer via the USB port.
2. Confirm the new port that's appeared. In Windows, you can check this by bringing up your Device Manager. On a MAC, you can check your System Profiler.



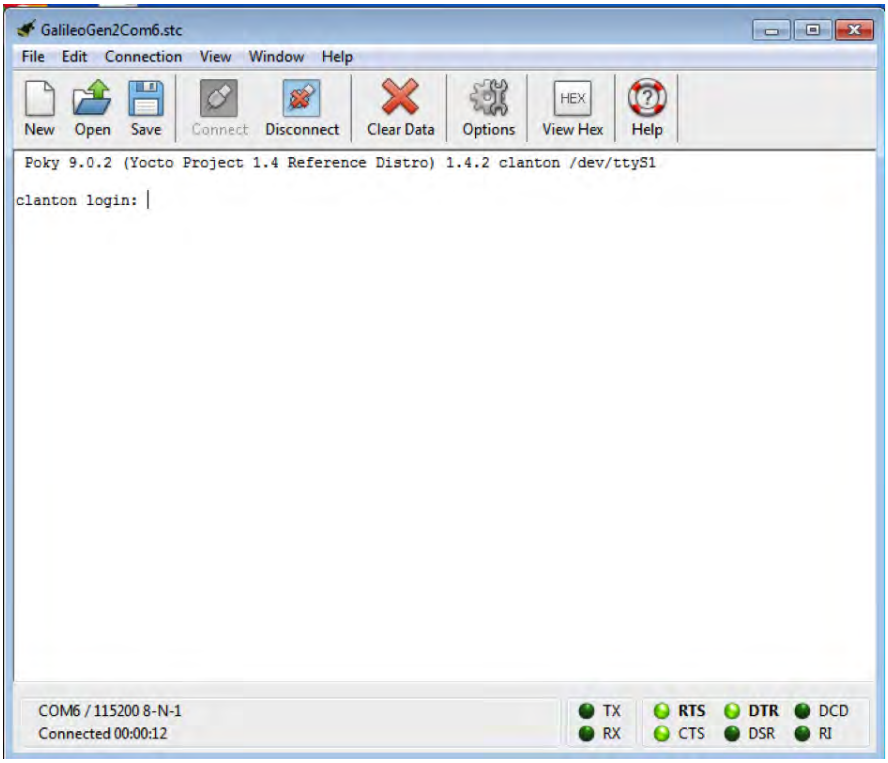
3. Launch CoolTerm and click Options.
4. For Baudrate select 115200.
5. Click Re-scan Serial Ports.
6. Select the Serial port you identified in Step 2.



7. Click Terminal on the left to show additional options.
8. Choose CR for Carriage Return Emulation.
9. Check the box for 'Handle Backspace Character' to enable it.



10. Go back to the main CoolTerm window.
11. Click Connect in the top toolbar and press Enter.
12. You should see a login prompt appear.



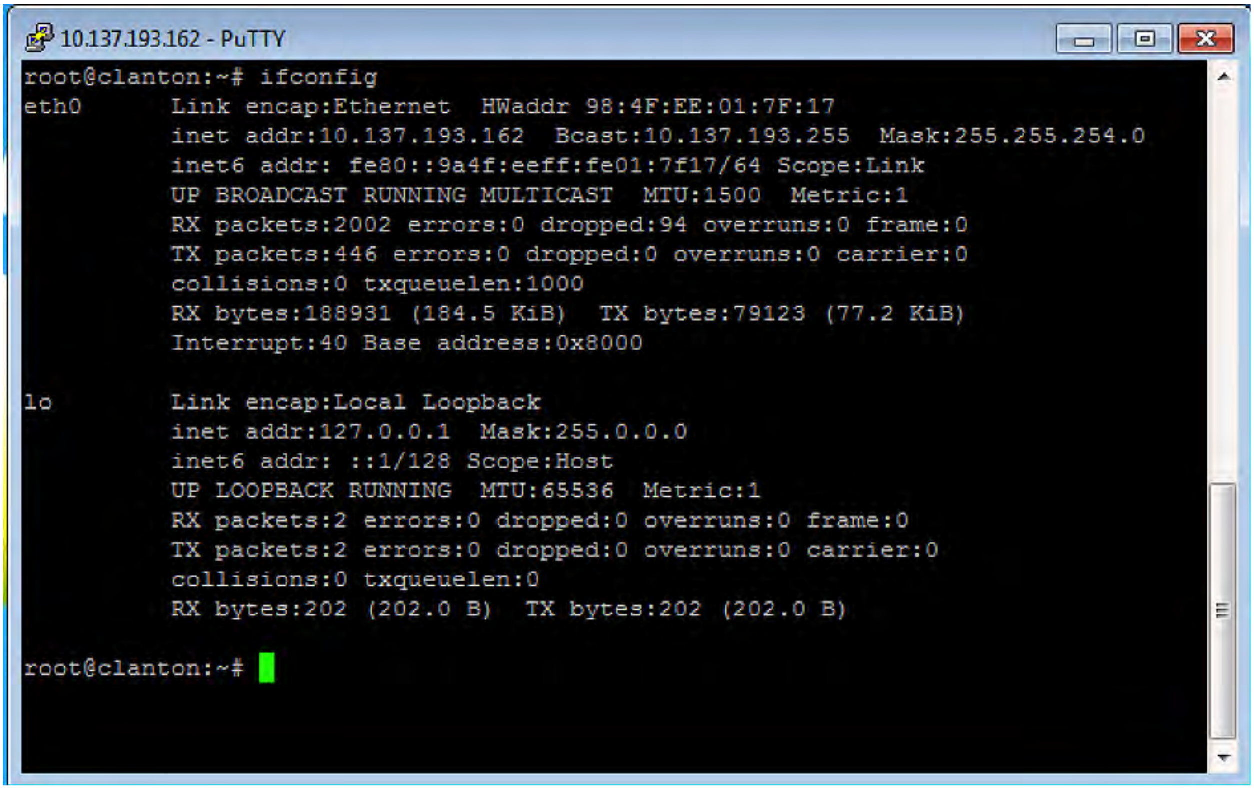
13. Login with user name root and press Enter.

You should be at your Galileo's Linux command line.

Linux and the Galileo

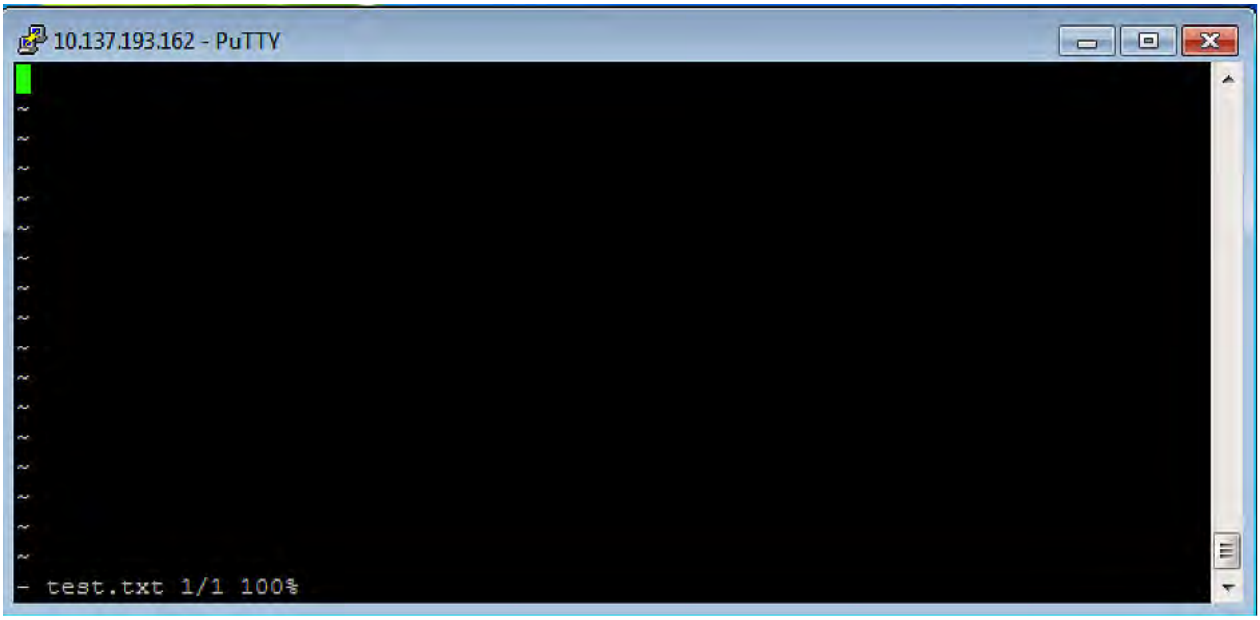
Once you have connected to the Galileo's Linux command line you can do a variety of useful things. You can list the contents of a directory using the `ls` command. You can change directories using the `cd` command. There is a great tutorial at http://linuxcommand.org if you want to learn more about the range of Linux commands available.

One really useful command is `ifconfig`. This will tell you your IP address just after the inet addr: in the output. You will need to know your IP address when you connect to your Galileo over the Ethernet.



Creating a program in Linux using vi

Linux comes with a built in editor called vi. In order to create a file called test.txt, for example, type at the command prompt `vi test.txt`



The vi editor will open and you will see a column of tildes(~). To add text press 'a' and start typing. When you want to save and quit type ":wq".

For more info on how to use vi see http://www.unix-manuals.com/tutorials/vi/vi-in-10-1.html

Resources

http://linuxsurvival.com

http://linuxcommand.org

Connecting via Ethernet

Connecting via an Ethernet Connection

Note: ***Some networks require additional configuration. See below for NSW DEC relevant instructions.***

1. Connect your Galileo via an Ethernet cable to your network.
2. Connect your Galileo to power.
3. Connect your computer to the Galileo via the USB client port.From the Arduino IDE select File>Examples>Ethernet>WebClient.
4. Modify the Arduino code to replace the MAC address with the one on your Galileo printed on top of the Ethernet port.
5. Upload your program.
6. Open the serial monitor by clicking on the magnifying glass icon on the top right of the screen, or selecting Tools from the menu bar and then 'Serial Monitor'.

If you have successfully connected you will see text start to appear in the Serial monitor. This program does a Google search for the term "Arduino". The HTML response is displayed on the serial monitor. You can turn off Autoscroll in your serial monitor and scroll back to the beginning.

The above program will not work on the NSW DEC network due to network restrictions. Follow the steps in the following section to access web pages on the internet.

Connecting to the NSW DEC network

The NSW DEC network has a two-step authentication system which makes it a bit more complicated to connect to. First, users must authenticate through a proxy and then they must authenticate to access each web site. Fortunately, we can use the `system()` function in combination with the linux '`curl`' command to retrieve and post data to and from the internet. The `system()` function allows us to run any Linux command from Arduino code.

```
system("curl --trace - --proxy1.0 proxyvip.cache.det.nsw.edu.au:8080 -U
myDECUserName:myPassword -u myDECUserName:myPassword http://www.dec.nsw.gov.au/ > /dev/
ttyGS0");
```

In the above call to the `system()` function from Arduino code, the Linux curl command is being called to retrieve and read a web page from the internet. The final part of the Linux command:

```
> /dev/ttyGS0
```

displays the output from the Linux command onto the serial console. You must replace myDECUserName:myPassword with your DEC user name and password.

The complete program you must download from your Arduino is as follows:

```
void setup() {
  Serial.begin(9600);
  delay(1000);
  system("curl --trace - --proxy1.0 proxyvip.cache.det.nsw.edu.au:8080 -U
myDECUserName:myPassword -u myDECUserName:myPassword http://dec.nsw.gov.au > /dev/ttyGS0");
}
void loop() {
  // do nothing
}
```

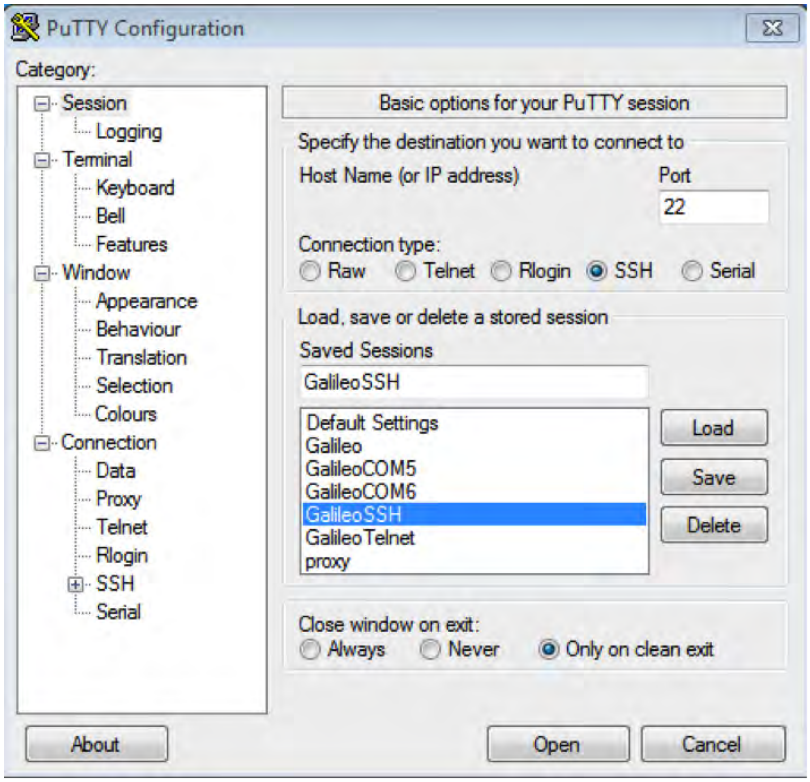
In the above program, the code is in the setup function as we only want to retrieve the contents of the web page once!

Connecting via SSH

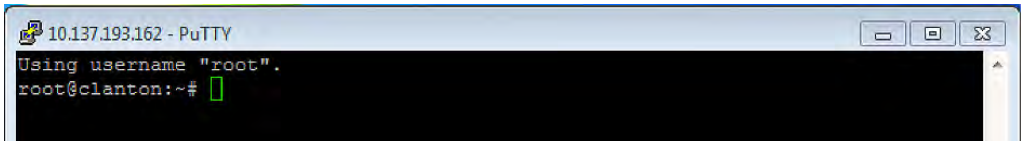
Once the Galileo is connected to the Ethernet then you can connect to the board using the Putty program if you are on Windows or the Terminal program on a Mac.

Connecting via Putty on Windows

1. Download Putty from www.putty.org
2. Start up Putty.
3. Select SSH as your connection type.
4. In the Host Name box enter root@Your_IP_Address and Port 22.
5. Click Open.

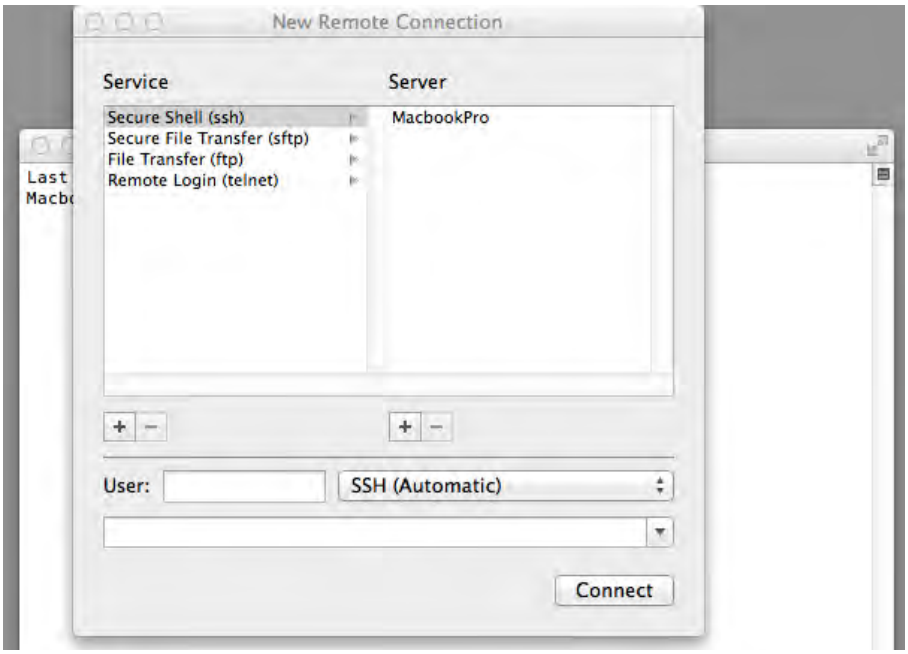


After clicking Open you should be presented with a Linux login prompt for your Galileo.

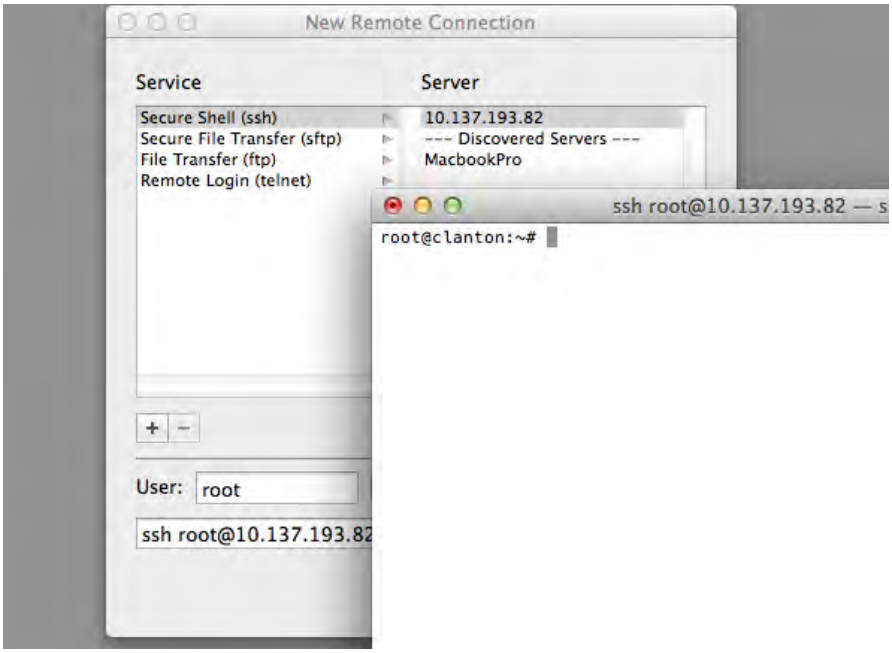


Connecting via Terminal on Mac OS X

1. Open the *Terminal* application, found in the *Utilities* folder under *Applications.*
2. From the 'Shell' menu, select 'New Remote Connection'.
3. Select 'Secure Shell (ssh)' under 'Service', then press the plus (+) symbol under 'Server'. Enter the Intel® Galileo's IP address in the box that pops up.



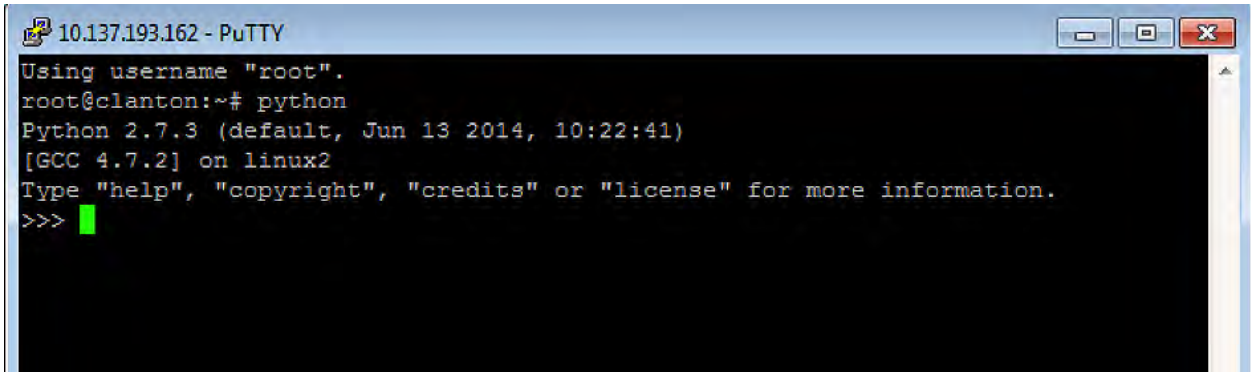
4. Type 'root' as the User.
5. Click 'Connect' , a new prompt will open displaying 'root@clanton:~#'



Python and the Galileo

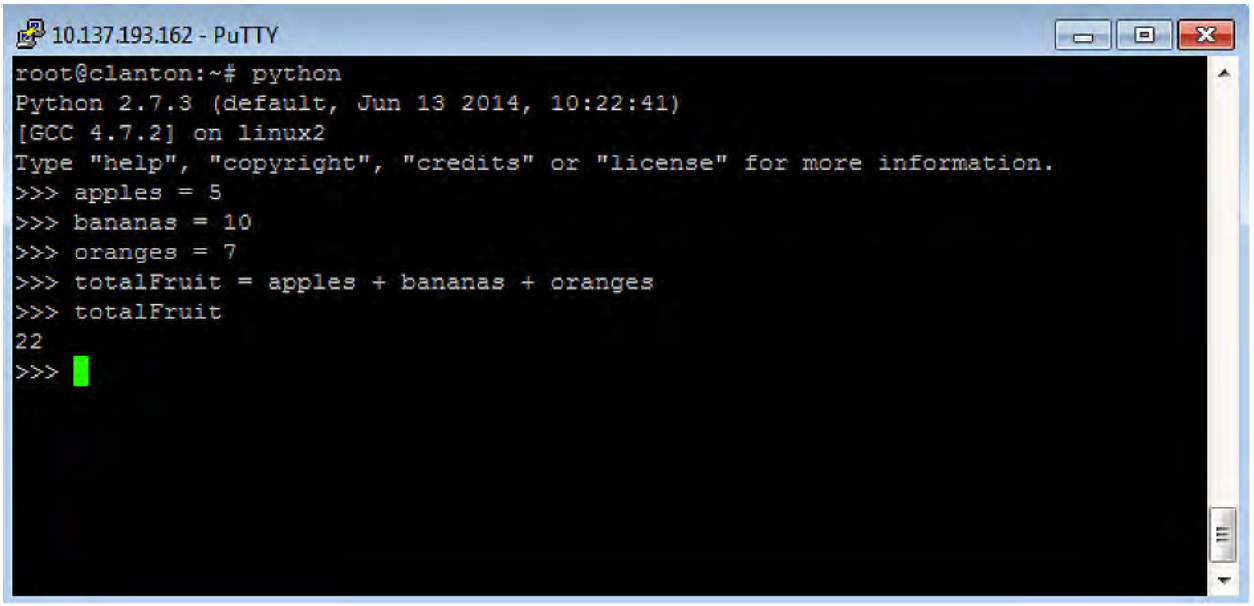
Python 2.7.3 comes installed on the Galileo Linux image which means you can use the Galileo to teach kids Python programming. In this section we will introduce a couple of key concepts in Python and the reader is provided with several resources for going further.

Python can be used in an interactive way by issuing commands in the Python prompt. If you type python at the Galileo's Linux command prompt Python will come up in interactive mode



Variables

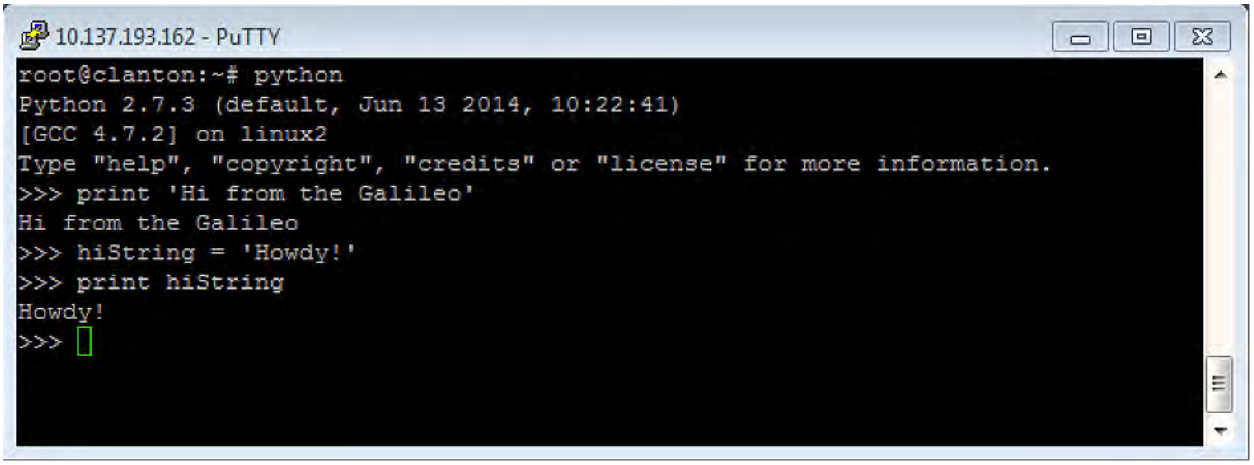
Variables are used in python to store values for later use and refer to them by name. Have a look at the following example which adds the different fruits. This example also shows that you can do calculations in Python. Try typing in different calculations at the command line and press enter instead of equals. The asterisk symbol is used for multiplication.

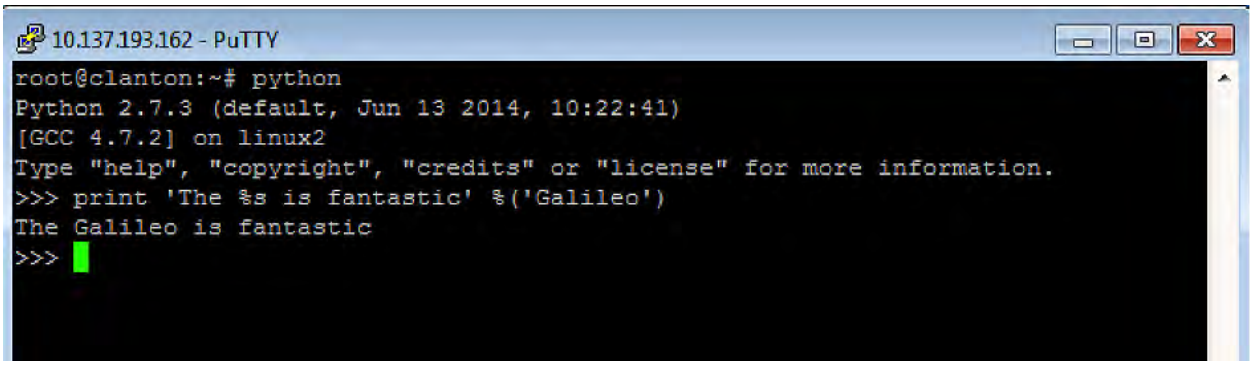


Strings

You can use Python to do all sorts of string printing, searching and manipulation. You can simply print a string or you can make a placeholder for a string in a variable and print out the variable.

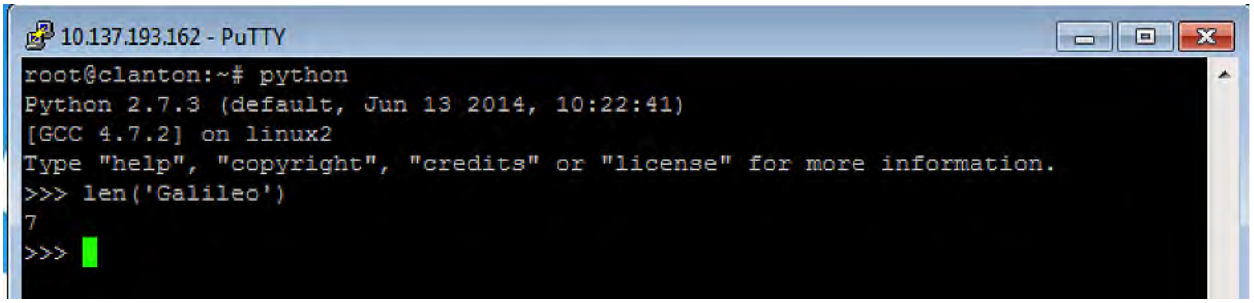
The %s allows you to insert variables into the middle of strings as in the example above.





Built in function len

Python also has many built in functions such as the len function which returns the length of a string.



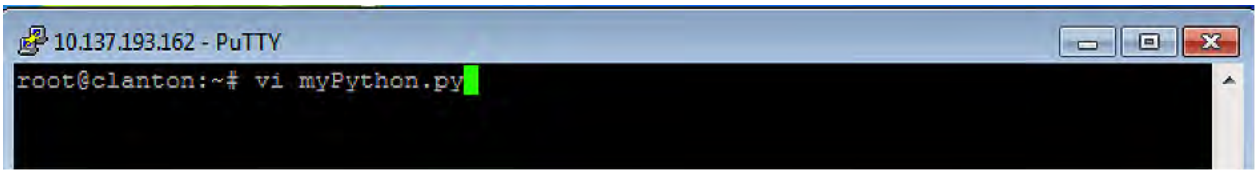
Modules in python

Python comes with loads of built-in modules for doing many common functions ranging from creating a Web Server to doing particular mathematical calculations. You can see the full list of built-in functions in Python here https://docs.python.org/2/library. To import one of these modules you simply type `import` and the name of the module usually in the first few lines of the program. You can even write your own modules.

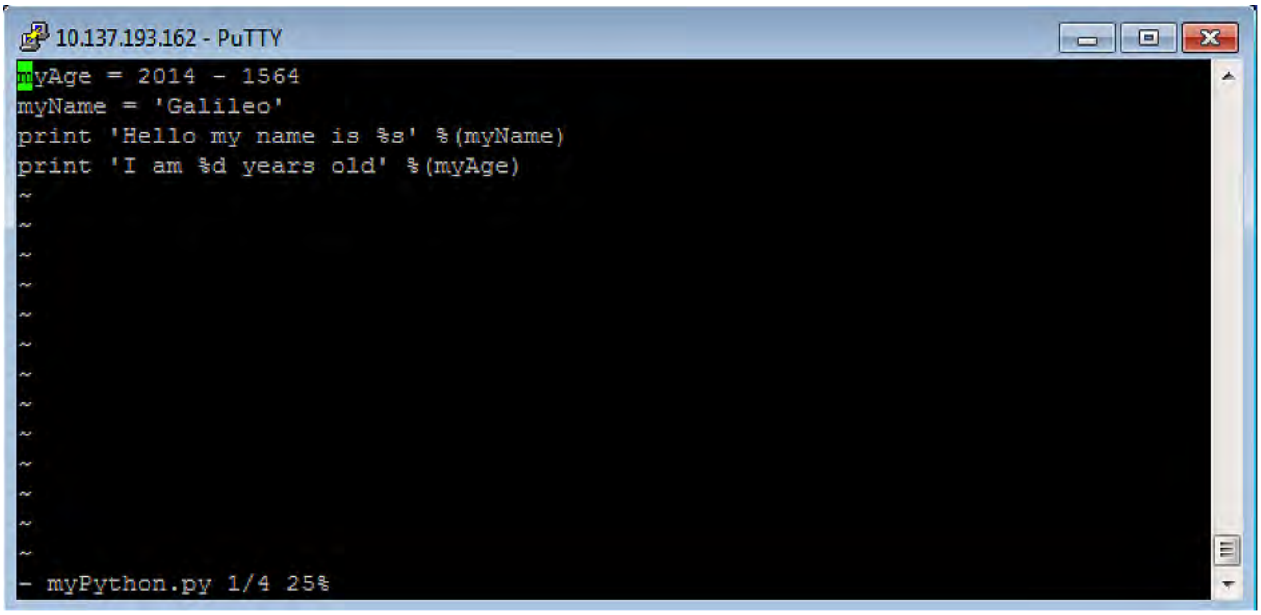
Running Python programs

In python you can store a collection of commands in a text file with extension.py and then run the whole program from the command prompt. These are run from the Linux command line by typing: `python fileName.py`

Create a new python program using vi called `MyPython.py` as in the screen shot. Begin by typing:



Then press 'a' to start adding content.



And type in :wq to save and quit. Then type `python myPython.py` to run the program.

Indentation

In Python, the way the program is formatted means something when the program is run. This is different from other languages like Java and C or C++. Code with the same indentation is considered to be in the same block. So in python there is no need to use an opening and closing brace to 'block' out some code. In python the number of whitespaces will determine which code goes with which.

Python Resources:

www.codeacademy.com
www.sthurlow.com
www.learnpython.org
http://inventwithpython.com/chapters/
http://python.org

Javascript and Node.js on the Galileo

Overview

This activity involves coding up a simple web server on the Galileo using node.js. There is no hardware involved.

Materials

- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead

Coding

As part of the standard Linux image supplied with the Intel® Galileo there is something called **node.js**. **Node.js** is an environment that allows you to build high performance and scalable network applications in the Javascript language. Programs in node.js are written in Javascript. Javascript is a programming language that is built into all the major web browsers and used to make web pages interactive. Javascript is often embedded in HTML using the <script> tag. There are plenty of pre-written Javascripts that you can copy and plug straight into your web page.

Here is a simple html file with some embedded Javascript:

```
<html>
 <head>
  <script type="text/javascript">
function functionOne() { alert('You clicked the top text'); }
function functionTwo() { alert('You clicked the bottom text'); }
  </script>
 </head>
<body>
 <p><a href="#" onClick=”functionOne();">Top Text</a></p>
 <p><a href="javascript:functionTwo();">Bottom Text</a></p>
 </body>
</html>
```

Copy and paste this code into a new program named: `SimpleJavascript.html`

Then double click on this file to open it in a browser.

The <script> tells the browser that the following code is javascript and likewise the </script> ends the javascript code.

The onClick="functionOne();" tells the browser to perform the code in functionOne when you click on the Top Text link. Similarly, the "javascript:functionTwo();" tells the browser to perform the code in functionTwo when you click the Bottom Text link.

Most people use node.js as a web application server. Here is a simple web server written in node.js which we can run on our Galileo.

```
// Load the http module to create a http server
var http = require('http');

// Configure the http server to respond with Hello World to all //requests
var server = http.createServer(
    function(request,response) {
        response.writeHead(200,{'Content-Type': 'text/plain'});
        response.end('Hello World\n');
  }
);

server.listen(1337, '192.168.1.18'); // IP address of the Galileo
console.log('Server running at http://192.168.1.18:1337/');
```

In order to run this code on our Galileo, in a Putty or Terminal window connected to your linux image create a new file by typing in at the command prompt:

```
root@clanton:~# vi  hello.js
```

(See the section of this document on Linux for more details about creating a file using vi.)

Type the program contents above into hello.js.

Now run the program by typing at the command prompt:

```
root@clanton:~# node hello.js
```

Now type in the address into your browser and there you have a simple web server visible on your local network. In our case we would type in: http://192.168.1.18:1337/

Node.js Resources:

http://www.crunchzilla.com/code-monster -- A great place for kids to learn Javascript
http://nodetuts.com
http://www.nodebeginner.org/

Displaying temperature data on a WebServer using Google Charts

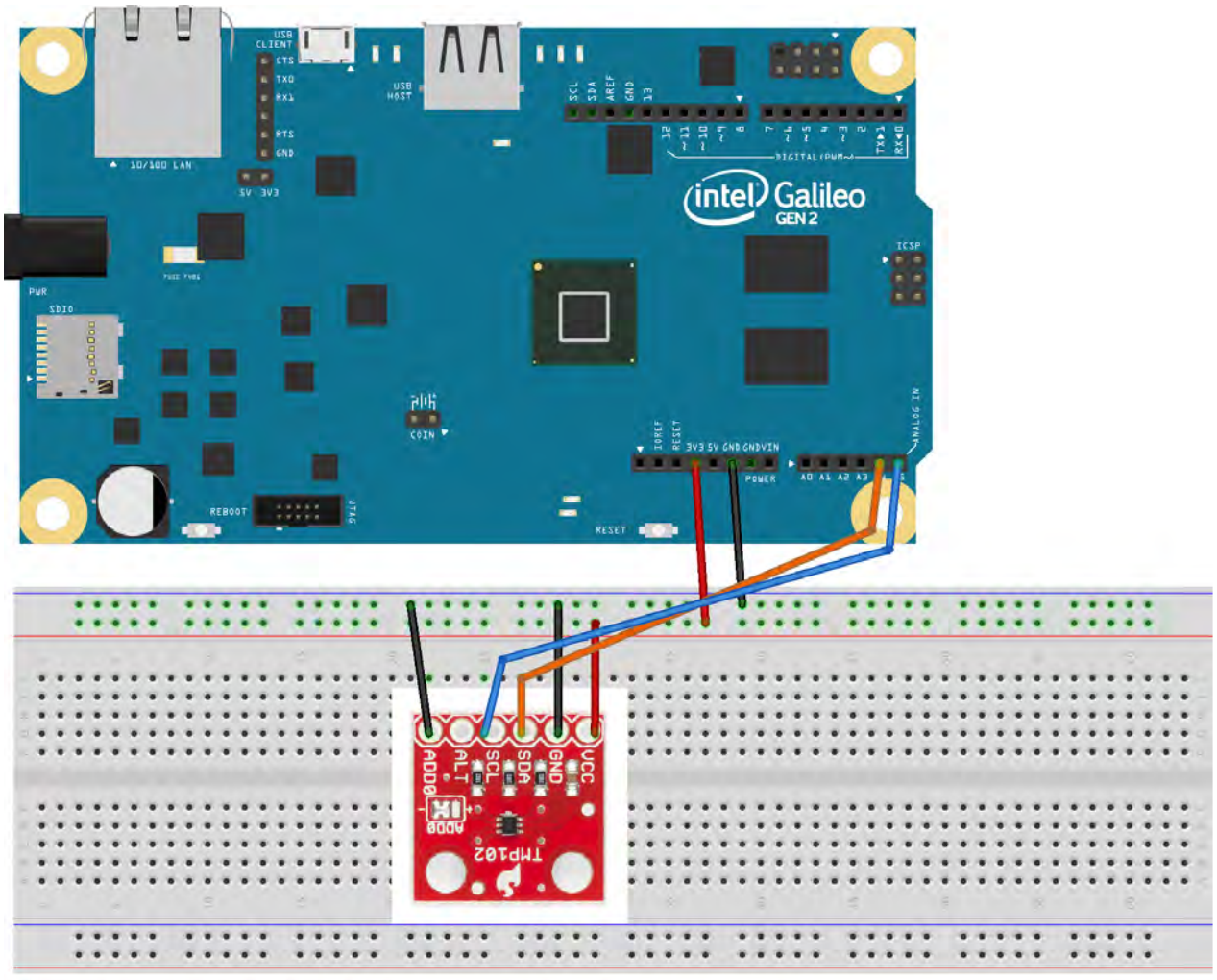
Overview

In this activity we connect a temperature sensor to the Arduino and continually write the current temperature to a file on the Galileo. We then run a python program on the Galileo to read the temperature on this file and display it on a web server using the Google Charts API. When the user refreshes the browser on the web server the current temperature is retrieved and displayed. This activity requires soldering so will not be suitable for younger students.

Materials

- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead
- 1 x Breadboard
- 5 x Jumper leads (M-M)
- 1 x Temperature Sensor TMP102
- 1 x Soldering Iron + solder

Wiring



Connect temperature sensor to the Galileo as follows:

1. Solder jumper wires to five of the six connectors on the TMP102. Leave the ALT connection clear.

For instructions on how to solder see http://www.wikihow.com/Solder-Electronics

2. Connect ADD0 pin to GND on the Galileo.
3. Connect VCC pin to 3.3V on the Galileo.
4. Connect SDA pin to Analog Pin 4 on the Galileo (A4).
5. Connect SCL pin to Analog Pin 5 on the Galileo (A5).
6. Connect GND pin on the TMP102 to GND on the Galileo.

Coding the Arduino

Upload the following sketch to your Arduino.

```
// This program reads a value from the TMP102 temp sensor on the
// Galileo and writes it to a file and the serial output for debugs.
////////////////////////////////////////////////////////////////

#include <Wire.h>
int tmp102Address = 0x48;
FILE* tempFile;

void setup(){
  Serial.begin(9600);
  Wire.begin();
}

void loop(){

  float celsius = getTemperature();
  Serial.print("Celsius: ");
  Serial.println(celsius);

  // create file for writing temperature to if it doesn't exist
  tempFile = fopen("/home/root/currentTemperature", "w+");

  // now write to file on the Galileo
  if (tempFile != NULL) {
      // convert float to a string
      fprintf(tempFile,"%.2f",celsius);
      fclose(tempFile);
  }
  else {
   Serial.println("File doesn't exist");
  }
```

```
  delay(200); //slow down the output.
}

float getTemperature(){
  Wire.requestFrom(tmp102Address,2);

  byte MSB = Wire.read();
  byte LSB = Wire.read();

  //it's a 12bit int, using two's compliment for negative
  int TemperatureSum = ((MSB << 8) | LSB) >> 4;

  float celsius = TemperatureSum*0.0625;
  return celsius;
}
```

Explanation of the code

The `getTemperature()` method uses pre-packaged code from the Wire library which allows it to read the value from the temperature sensor. Then some binary maths has to be performed to return a temperature in Celsius.

Once the temperature is read it is then converted to a string and written to a file. The fopen() command opens a file for writing and if the file exists it will write over it as we only want to store the current temperature in the file.

The Arduino code should be writing the current temperature to a file called currentTemperature in your home directory on the Galileo. Open your Linux console using Putty or Terminal and confirm the file is present and reflecting the current temperature.

Creating the Python Web Server

A python program is created to host a web server on the Galileo which displays the current temperature from the file written to by the Arduino.

Create the following python program, tempServer.py, using the vi editor (See section on Linux and vi) and run this code from the Linux command line as follows:

```
root@clanton:~# python tempServer.py

import SocketServer
import SimpleHTTPServer

PORT = 8081
currentTemp = 0
class MyTCPServer(SocketServer.TCPServer):
    allow_reuse_address = True
class HandleRequests(SimpleHTTPServer.SimpleHTTPRequestHandler):

    def do_GET(self):
```

```
        # print HTTP header first
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()

        lines = []
        tempFile = open("currentTemperature","r")
        lines = tempFile.readlines()
        global currentTemp
        currentTemp  = float(lines[0])
        print 'Temp is ' + lines[0]

        html = """
        <html>
        <head>
        <script type='text/javascript' src='https://www.google.com/jsapi'></script>
        <script type='text/javascript'>
        var currentTemp;
        var data;
        var options;
        var chart;
        google.load('visualization','1.0',{'packages':['gauge']});
        google.setOnLoadCallback(initChart);
        function initChart() {
        data = new google.visualization.DataTable();
        data.addColumn('string', 'Label');
        data.addColumn('number', 'Value');
        data.addRows(1);
        data.setValue(0,0,'Temperature');
        data.setValue(0,1,%s);
        options = {width: 300, height: 300, greenFrom: 0, greenTo: 50, redFrom: 75, redTo:
100, yellowFrom:50, yellowTo:75, minorTicks:5};
        chart = new google.visualization.Gauge(document.getElementById('gauge_div'));
        chart.draw(data, options);
        }
        </script></head>
        <BODY style='font-family:Arial;'>
        <div id='gauge_div'></div>
        </body></html>
        """ %(currentTemp)

        self.wfile.write(html)

httpd = MyTCPServer(("",PORT), HandleRequests)
print "Serving from port", PORT
httpd.serve_forever()
```

Explanation of the code

The code for this web server makes use of two different libraries with built in functions for web servers: the SocketServer library and the SimpleHTTPServer library. The MyTCPServer class makes use of the SocketServer library while the HandleRequests class makes use of the SimpleHTTPServer library.

In this code we basically create a MyTCPServer serving at port 8080 and then call the serve_ forever method. This method will cause the do_GET method to be called whenever there is a GET request from the browser. Thus whenever the user refreshes the browser this code will be called.

Inside the do_GET method, the temperature is read from the currentTemperature file into a global variable called currentTemp.

Then a multiline string is formed for the html for the page with the current temp being substituted in for the %s inside the html. The rest of the code is Javascript for creating the Google Chart which calls on Javascript code within the browser. See below for tutorials on how to implement Google Charts.

Starting the Web Server

Open up a browser and go to the web address http:XX.XX.XX.XX:8081 (where XX.XX.XX.XX is your IP address of your Galileo). Refresh the browser to display the latest ambient temperature from your Galileo.

Going Further

You can use Google Charts to plot all sorts of different charts. For example, we could write temperature every 10 minutes for 12 hours overnight into a file on the Galileo and then plot a night's worth of temperature on our web server as a Line Graph using Google Charts. You could also use the Galileo to record soil moisture readings and then plot them using Google Charts.

Resources on Google Charts

https://developers.google.com/chart/interactive/docs/quick_start

https://developers.google.com/chart/interactive/docs/examples#gauge_example

http://code.tutsplus.com/tutorials/easy-graphs-with-google-chart-tools--net-11771

CONNECTING TO THE OUTSIDE WORLD

Note: ***At the time of this guide's publication, it was not possible for the following three projects to be carried out using the NSW DEC network as there were not sufficient ports open for communication.***

Sending Email from your Galileo

Overview

This activity sends an email from your Galileo when you run the program. It makes use of the Simple Mail Transfer Protocol (SMTP) to send the email. The SMTP protocol is the protocol used by the majority of all mails that are transferred around the internet. Here are a couple of links about how SMTP works:

http://www.ballisticdomains.com/articles/art-web-hosting/138-smtp-servers
http://www.laynetworks.com/smtp.htm

Materials

- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead

Coding

1. Connect up to your Galileo using Putty or Terminal.
2. From the Linux console, using the vi editor, copy and paste the following code into a file called SendEmailOnButtonPress.py
3. Run the program from the command prompt by typing in:
4. root@clanton~# python SendEmailOnButtonPress.py
5. Press the button and an email should be sent.

```
import smtplib

FROM='SarahsGalileo@gmail.com' # can be a fake email address

TO=[recipient1@email.com, recipient2@email.com,]

SUBJECT="I am alive"

TEXT="This is from the Galileo"

#Prepare actual message
message = """\
From: %s
To: %s
Subject: %s

%s
```

```
""" % (FROM, ", ".join(TO), SUBJECT, TEXT)

# Send the mail
server=smtplib.SMTP('mail.server.com')  #your email server address
server.sendmail(FROM, TO, message)
server.quit()
```

Explanation of the code

The first line of the program imports some code from another module, smptlib, which knows all about sending email.

```
import smtplib
```

Then the next few lines create some variables to represent certain aspects of the email message.

```
FROM='SarahsGalileo@gmail.com' # can be a fake email address

TO=[recipient1@email.com, recipient2@email.com,]

SUBJECT="I am alive"

TEXT="This is from the Galileo"
```

The following lines prepare the actual message using a multi-line string which is indicated by the starting and ending triple quotes “””\.

```
message = """\
From: %s
To: %s
Subject: %s

%s
""" % (FROM, ", ".join(TO), SUBJECT, TEXT)
```

Anywhere python sees a %s sign inside the multi-line string it will replace it with the corresponding string from the tuple at the end of the multiline string prefaced by the %.

```
""" % (FROM, ", ".join(TO), SUBJECT, TEXT)
```

The final three lines use the smptlib functions to send the email message.

```
server=smtplib.SMTP('mail.server.com')  #your email server address
server.sendmail(FROM, TO, message)
server.quit()
```

You must replace `mail.server.com` with your email server address.

Checking Email from your Galileo

Overview

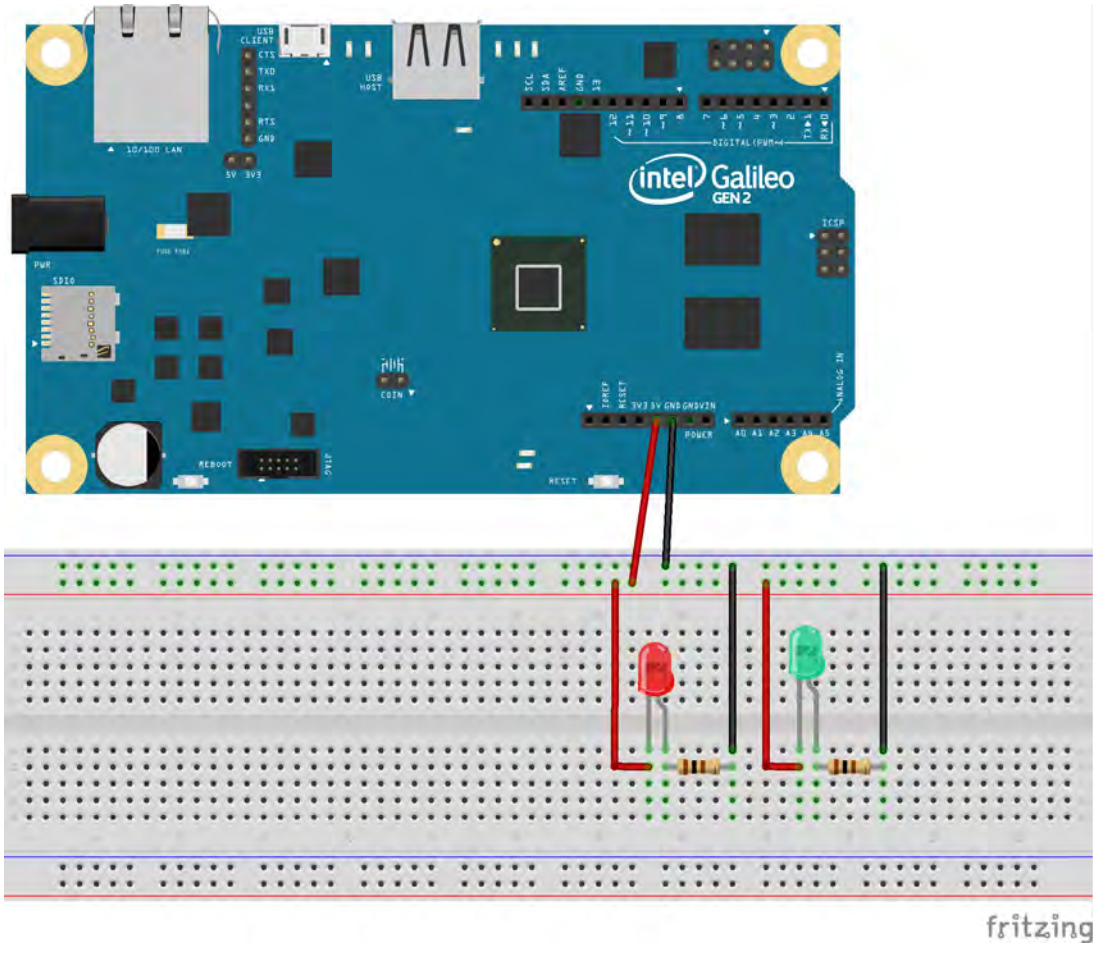
This activity checks email from your Galileo and lights up an LED when you have unread email. This was adapted from a more complicated project from Sparkfun which you can find at https://learn.sparkfun.com/tutorials/galileo-unread-email-counter It is issued under a Creative Commons BY-NC-SA 3.0 licence.

Materials

- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead
- 1 x Breadboard
- 1 x LED
- 3 Jumper Wires (M-M)
- 2 x 100 Ω Resistors

Wiring

Using the breadboard, wire up the negative (shorter) lead of the Red LED to a resistor and wire up the other end of the resistor to the ground on the breadboard. Wire up the positive (longer) lead of the LED to Pin 13. This LED will indicate if you are connected to the internet. Similarly, wire up the negative (shorter) lead of the green LED to a resistor and wire up the other end of the resistor to the ground on the breadboard. Wire up the positive (longer) lead of the green LED to Pin 12. This LED will indicate if you have unread emails.



Coding the Python

Create a python file called UnreadEmailChecker.py with the following contents:

```
import imaplib
obj = imaplib.IMAP4_SSL('imap.gmail.com', '993')
obj.login('yourEmail@gmail.com,'yourPassword')
obj.select()

print len(obj.search(None,'UnSeen')[1][0].split())
```

Explanation of the code

The first line imports a module called imaplib which has functions for sending mail.

```
import imaplib
```

The next three lines initialise an object with the mail server address and login to your mail server . You need to put in the correct details here for your mail server and your mail account.

```
obj = imaplib.IMAP4_SSL('imap.gmail.com', '993')
obj.login('yourEmail@gmail.com,'yourPassword')
obj.select()
```

The final line uses the search method to return the number of UnSeen emails and then prints a count of these. For more information on the imaplib methods go to https://docs.python.org/2/library/imaplib.html

```
print len(obj.search(None,'UnSeen')[1][0].split())
```

Breaking this line down further, `obj.search(None,'UnSeen')` returns a tuple of which the second element has data and we are interested in the first item of data which is why we have the array index [1][0]. The response is a space separated list so we need to call the split() command to remove the final space.

The len() command gives us the length which corresponds to the number of emails and then we print that number.

Coding the Arduino

Upload the following sketch to your Arduino and you should start getting your light going on when you have unread emails.

```
/* Galileo Email Checker
   by Sarah Boyd
   Adapted from Jim Lindblom
       SparkFun Electronics
   This code runs a Python script which calls a Gmail account and
   writes the number of unread emails to a file.
*/
//////////////////////
// Library Includes //
//////////////////////
#include <Ethernet.h> // Ethernet library

//////////////////////////
```

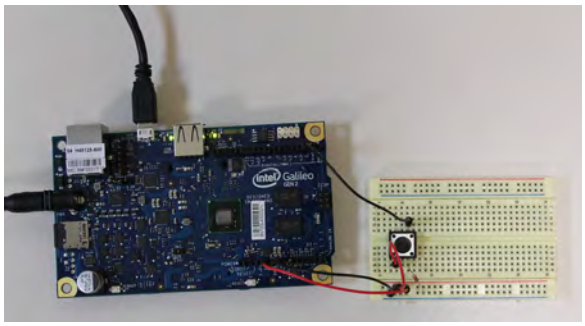
```
// Ethernet Definitions //
//////////////////////////
byte mac[] = {0x??, 0x??, 0x??, 0x??, 0x??, 0x??}; // put your Galileo MAC address here

//////////////////////
// Pin Definitions //
//////////////////////
const int STAT_LED = 13; // The Galileo's status LED on pin 13
const int UNREAD_LED = 12; // Lights up when there are unread emails

///////////////////////////////////
// Email-Checking Global Variables //
///////////////////////////////////
const int emailUpdateRate = 10000; // Update rate in ms (10 s)
long emailLastMillis = 0; // Stores our last email check time
int emailCount = 0; // Stores the last email count

void setup()
{
  pinMode(STAT_LED, OUTPUT);
  pinMode(UNREAD_LED, OUTPUT);
  digitalWrite(STAT_LED, LOW);
  digitalWrite(UNREAD_LED, LOW);

  Serial.begin(9600); // Serial monitor is used for debug

  if (Ethernet.begin(mac) == 0) {// If Ethernet connects, turn on Stat LED
    digitalWrite(STAT_LED, HIGH);
    Serial.println("Connected to the Ethernet OK.");
  }
  else
  { // If Ethernet connect fails, print error, inifinite loop
    while (1)
      ;
  }
}

// loop() checks for the unread email count every emailUpdateRate
// milliseconds. If the count has changed, update the display.
void loop()
{
  // Only check email if emailUpdateRate ms have passed
  if (millis() > emailLastMillis + emailUpdateRate)
  {
    emailLastMillis = millis(); // update emailLastMillis

    // Get unread email count, and store into temporary variable
    int tempCount = getEmailCount();
    Serial.print("You have ");
    Serial.print(tempCount);
```

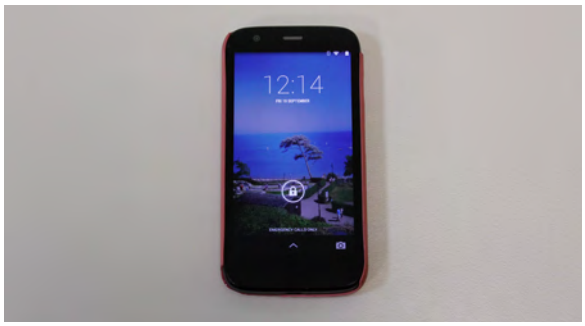
```
    Serial.println(" unread mail(s).");
    if (tempCount != emailCount) // If it's a new value, update
    { // Do this to prevent blinking the same #
      emailCount = tempCount; // update emailCount variable
      digitalWrite(UNREAD_LED, HIGH); // turn the LED on
    }
    else if (tempCount == 0){
      digitalWrite(UNREAD_LED, LOW);
    }
  }

  // Bit of protection in case millis overruns:
  if (millis() < emailLastMillis)
    emailLastMillis = 0;
}

// getEmailCount runs the unReadEmailChecker.py script, and reads the output.
// It'll return the value printed by the unReadEmailChecker.py script.
int getEmailCount()
{
  int digits = 0;
  int temp[10];
  int emailCnt = 0;

  // Send a system call to run our python script and route the
  // output of the script to a file.
  system("python /home/root/decUnReadEmailChecker.py > /dev/ttyGS0");

  char line[100];
  FILE* emailsFile = fopen("/home/root/emails", "r");
  if (emailsFile) {
      // read from emails until we hit the end or a newline
      while (fgets(line, 100, emailsFile) != NULL) {
        ;
      }
      fclose(emailsFile);
  }
  else {
    Serial.println("No emails file created");
  }
  temp[digits++] = atoi(line); // convert string to an integer

  system("rm emails"); // remove the file

  // Turn the individual digits into a single integer:
  for (int i=0; i<digits; i++)
    emailCnt += temp[i] * pow(10, digits - 1 - i);

  return emailCnt;
}
```

Sending texts from your Galileo

Overview

This activity sends a text from your Galileo to a connected mobile phone when you press a button, using the pushingbox service. Once again a pull down resistor is used to prevent the pushbutton from being in a floating state when not pressed.



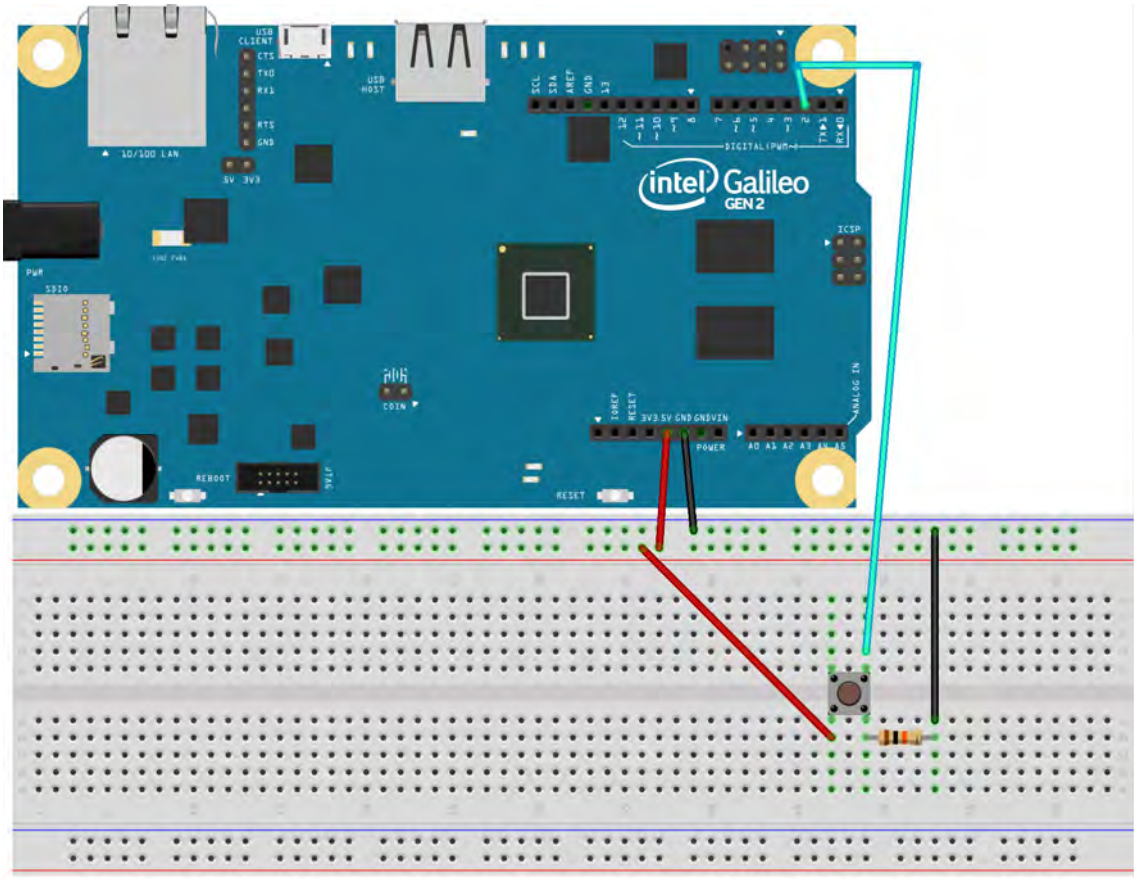


Materials

- 1 x Intel® Galileo Gen 2 board
- 1 x 12V Power supply
- 1 x Micro USB lead
- 1 x Breadboard
- 4 x Jumper wires (M-M)
- 1 x 10K Resistor
- 1 x Button

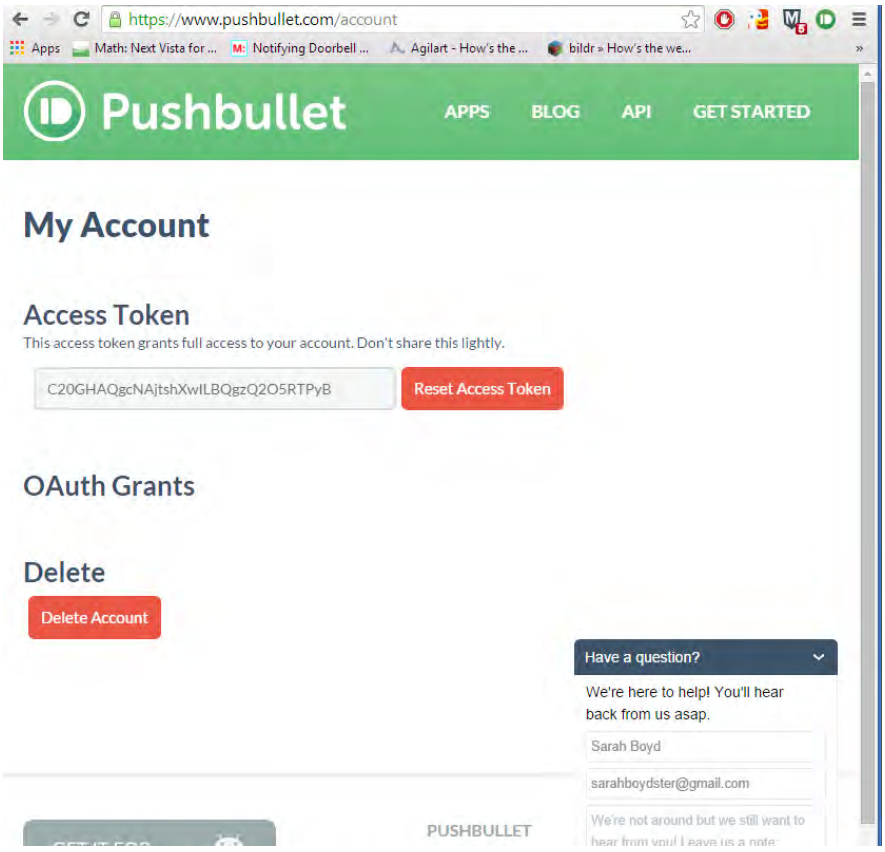
Wiring

1. Place the momentary push button in the middle of the breadboard with the divider underneath.
2. Connect a wire from the 5V pin out on the Galileo to the power strip on the breadboard.
3. Connect a wire from the GND pin out on the Galileo to the ground strip on the breadboard.
4. Connect the left-bottom side of the push button to the positive strip on the breadboard.
5. Connect the right-bottom side of the push button to one side of the resistor.
6. Connect the other side of the resistor to the ground strip on the breadboard.
7. Connect a wire from the top right side of the push button to pin out 2 on the Galileo.

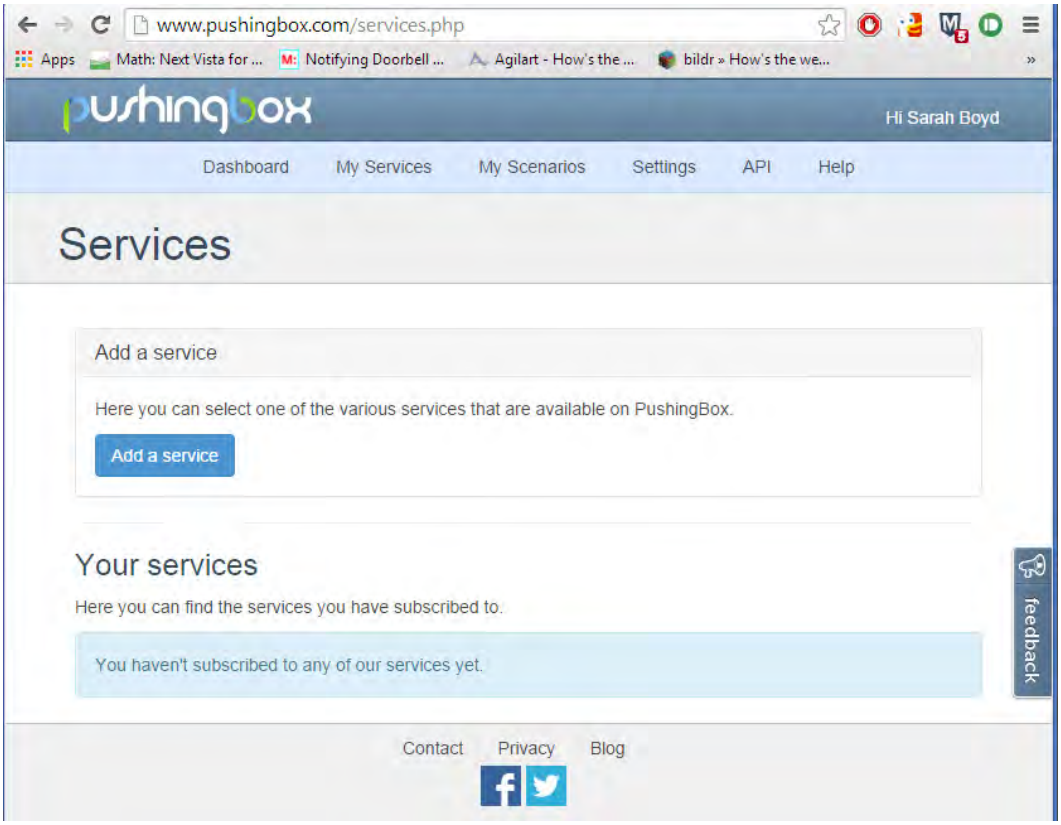


Setting up the pushingbox service

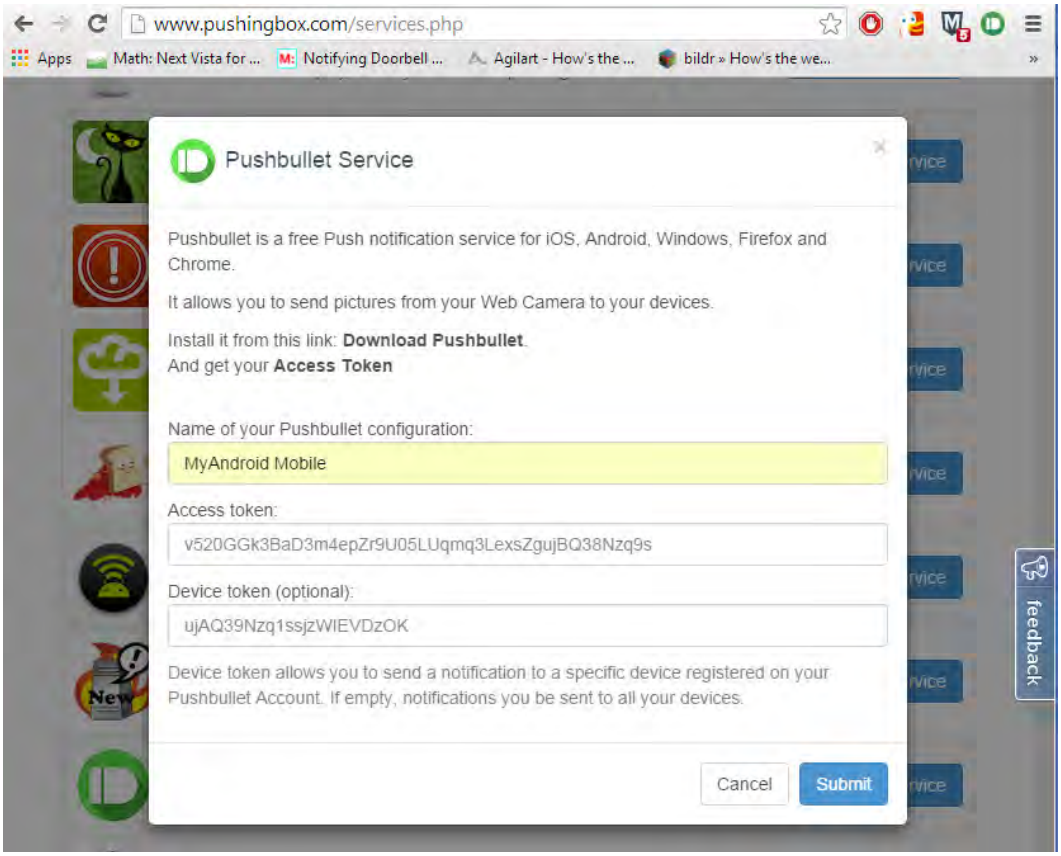
1. Download Pushbullet to your mobile phone and link to your Google gmail address.
2. Go to https://www.pushbullet.com/account and copy the Access token.



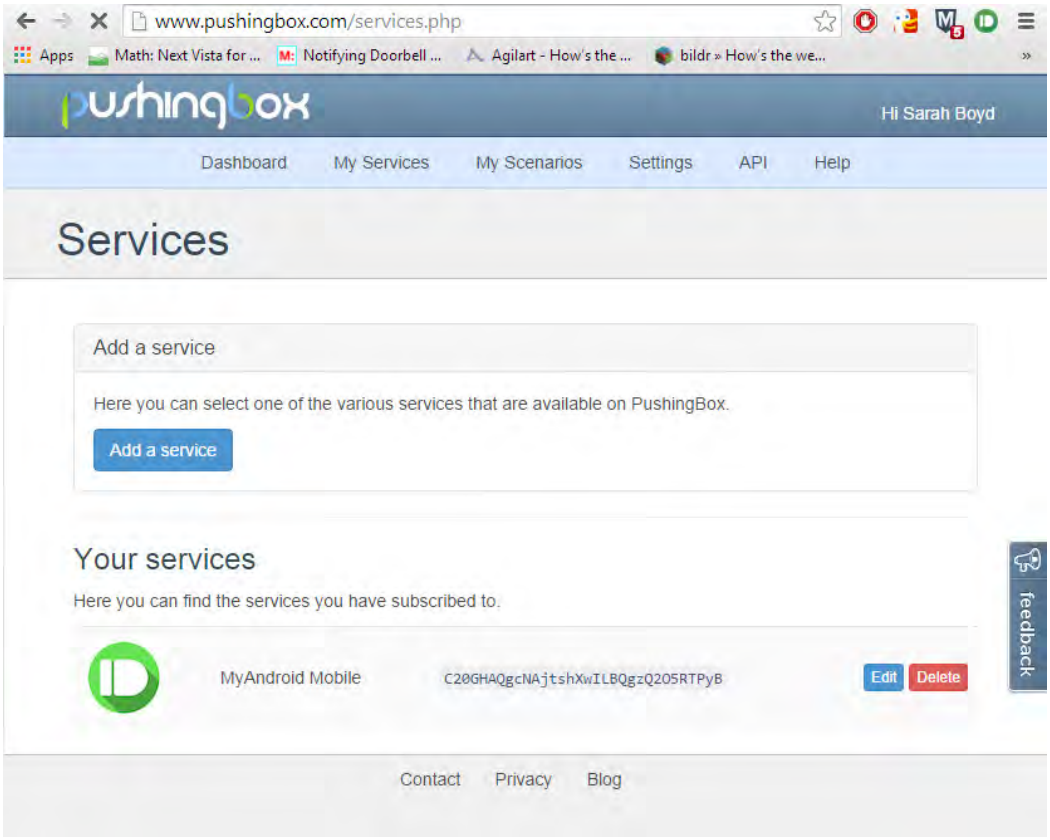
3. Create a pushingbox account at http://www.pushingbox.com
4. Login with your Google account.
5. Go to the MyServices page.



6. Click on Add service and select Pushbullet.

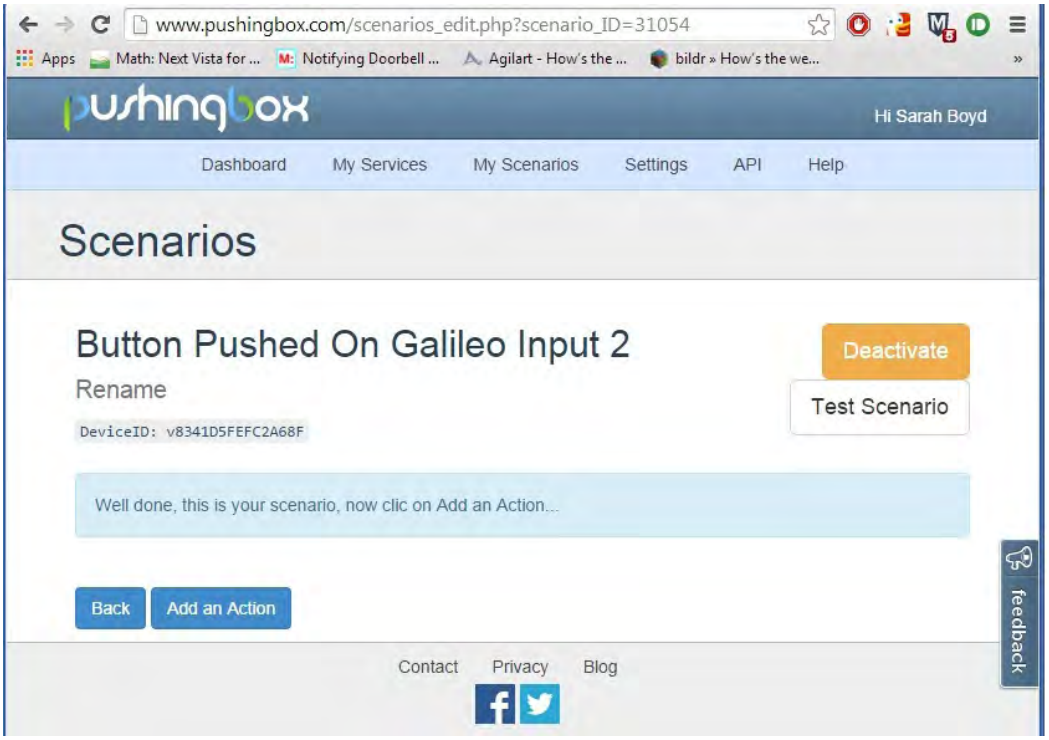


7. Name your Pushbullet configuration whatever you like. I called mine MyAndroidPhone.
8. Copy in your Access token you got from your pushbullet account settings in Step 2.
9. Leave the Device token blank and press Submit.

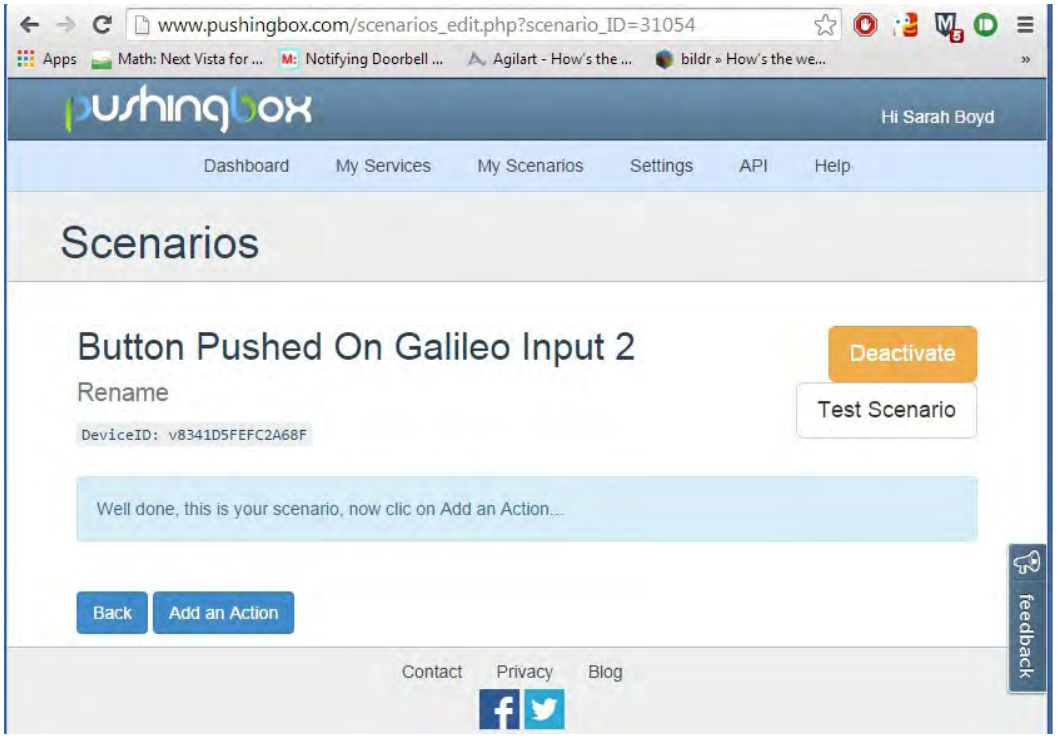


Now we need to create a scenario

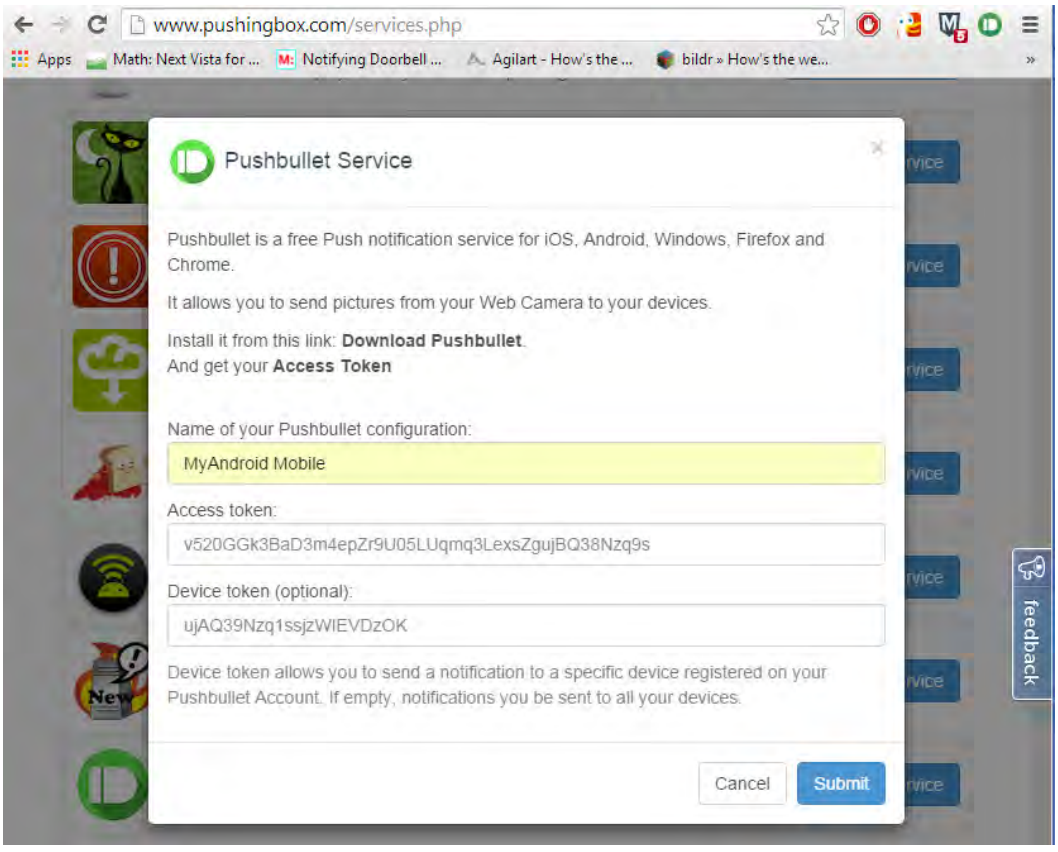
10. Go to My Scenarios.
11. Enter the name of your scenario and press Add.



12. Now click on Add an Action and select the Pushbullet service as an action.

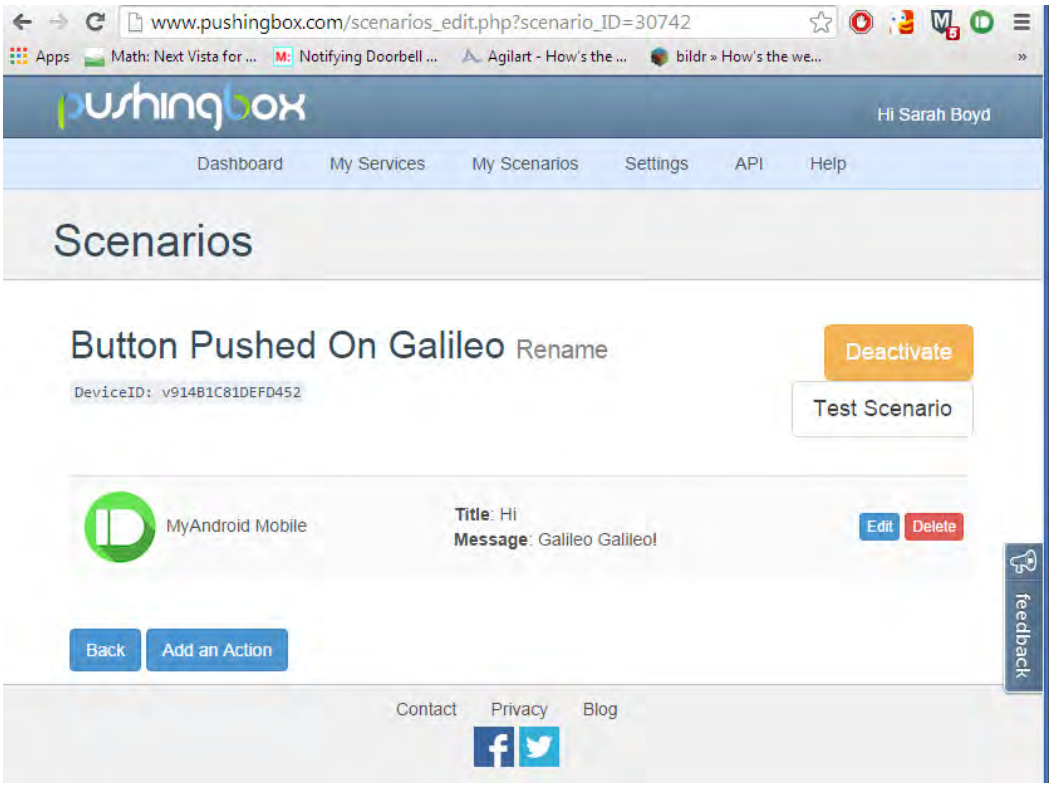


13. Now add an action with this service.



14. You can type in the title, text and optionally a photo for your message and then click Submit.

15. Click on the Back button to return to the Scenario page.



16. You can now test your scenario by clicking on Test Scenario.
17. You should receive a mobile phone message.

When you call pushingbox with the DeviceId of this scenario, you will prompt the associated action to occur. In our case, to send a text message to our connected mobile. We will use our Arduino code to call pushingbox.

The Arduino program

The Arduino program simply makes a call to the pushingbox service. Upload the following sketch:

```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(2, INPUT); // connect your switch between Pin 2 and 5V
  pinMode(12, OUTPUT); // connect an LED so you know when you are sending
}

void loop() {
  // put your main code here, to run repeatedly:
  int sensorValue = digitalRead(2);
  digitalWrite(12, sensorValue);

  if (digitalRead(2) == HIGH) {
    system("curl 'http://api.pushingbox.com/pushingbox?devid=vXXXXXXXXXXX' > /dev/
ttyGS0"); // put your devid instead of XXXX
  }
  delay(500);
}
```

GLOSSARY

6-pin FTDI header	Pins on the Galileo for connecting the serial cable.
ArduBlock	A software plugin for the Arduino IDE to allow visual programming using the Galileo.
Arduino	An Arduino is a development board that comes with its own IDE for downloading programs onto it.
Arduino IDE	The Arduino Integrated Development Environment (IDE) for writing and downloading software programs to Arduino compatible boards.
breadboard	A plastic board with slots for electronic components and wiring such that it is easy to wire up circuits.
COM port	A serial communications port on your computer.
device driver	A piece of software containing code to allow a particular device to work on your computer.
expansion pins	The pins on a Galileo that you can connect inputs and outputs to.
flash memory	A rewritable memory chip used for storage.
Galileo Gen 2	The second generation Galileo as opposed to Galileo Gen 1 which is the first generation Galileo.
Get request	An HTTP command used to retrieve data from a Web server.
GND	Electrical Ground.
Google Charts	A Javascript library of charting functions embedded into most modern browsers.
HTTP	HyperText Transfer Protocol. The communication protocol used to connect to Web servers on the internet.
Javascript	A web programming language which can be embedded in HTML and run in any web browser.
LED	Light Emitting Diode which emits light when electricity flows through it one way.
Linux	Open source computer operating system based on Unix.
Linux image	Refers to the Linux operating system and installed applications.
M-M jumper wire	Male to Male jumper wire.
MAC address	The unique serial number that identifies a network device.
method	A section of programming code that performs some specific function.
micro USB	The very small USB port found on the Galileo Gen 2 and also found on cellphones.
microSD Card	The smallest Secure Digital flash memory card.

mini PCI Express	A small version of the PCI Express peripheral interface for laptop computers. The WiFi card for the Galileo plugs into this interface.
multiline string	A string that runs over several lines of programming code.
Node.js	An environment that allows you to build applications in Javascript.
port	A jack or socket for connecting a network device.
potentiometer	A potentiometer provides a variable resistance output.
program	A computer program is a series of instructions that will be performed by the computer.
PushBullet	A mobile app that connects various network devices.
pushingbox	A cloud based service that can send notifications over the internet.
Putty	A free program for communicating between devices using various protocols such as SSH and serial communications.
Python	A popular programming language.
serial	Serial protocol. Transmitting data one bit at a time.
servo	A motor that moves a rotating arm to a particular angular position up to 180° when activated by electronic pulses.
sketch	A program that runs on Arduino compatible development boards.
SMTP	Simple Mail Transfer Protocol. A protocol for sending email messages.
SSH	Secure Shell. A secure protocol for connecting to a remote server.
USB to TTL serial cable	The cable for connecting serially with the Galileo Gen 2 and allowing you to access the Linux command line.
Web Server	A computer that runs a web site. Using HTTP the Web server 'serves' web pages to browsers as well as other web clients.

ARDUINO REFERENCE

For a comprehensive explanation of all the Arduino commands the Arduino Language Reference can be found at http://arduino.cc/en/Reference/HomePage

MacICT
Macquarie ICT Innovations Centre