Powering the Intel Galileo

Start by plugging the Galileo’s main power to its 5V input. An LED on the bottom left corner will light indicating that it is receiving power.

Next, plug in the micro USB cable to the USB client on the board.
**Important:** Always plug the Power cable BEFORE the USB cable or problems with your computer and/or the board could occur. When disconnecting, remove the USB cable before removing the Power for the same reason.

### A first program on the Intel Galileo

This program blinks an LED that is connected to pin 13. 

1. Prepare the board for the upcoming program by adding an LED to Pin 13.
   
   a. Remove power from the board (note: Disconnect the USB cable before the main power cable, never the other way around).

   b. Add a 330Ω resistor and LED in series between Pin 13 and ground as shown. Orient the LED properly, with the negative (short) leg plugged into ground. (The resistor limits the current flow, extending the LEDs life and keeping the microcontroller cooler).

2. In the Arduino IDE, navigate to File -> Examples -> 01.Basics -> Blink and you
will see a new sketch in a new window pop up.

3. Select “Verify” to compile the code.

4. Power the Galileo and plug in the USB cable.

5. Make sure the correct Board and Port is selected under “Tools”.

NOTE: It could take up to 30 Seconds for the Arduino IDE to recognize your board. If it is not recognized after 1 minute, proceed to manually rebooting the board by disconnecting the USB cable followed by the main power cable. Reconnect to try again.

6. Select “Upload” and the Led should start blinking at a rate of 1 second.
A program using a button for input

This program, reads an input button and sets two LEDs to 01 (if not pressed) or 10 (if pressed).

1. Prepare the board for the upcoming program. Be sure power is removed. Add an LED to Pin 12, and add a button connecting to ground on one end and to A0 on the other. When pressed, the button forms a connection; else there is no connection.

   Note: A0 will be programmed in pull-up mode, meaning that when the pin has no input the program will read it as 1, and when the pin has a 0 input (ground) the program will read it as 0. Note how the above button provides A0 with either no input (when not pressed, so read as 1) or with 0 (when pressed, so read as 0). Therefore, pressing the button causes A0 to be read as 0, and releasing as 1.

2. Create a new project named "lesson_button" with the following program:

   ```
   const int buttonPin = A0;     // the number of the pushbutton pin
   const int ledPinOne = 13;     // the number of the first LED pin
   const int ledPinTwo = 12;     // the number of the second LED pin
   ```
// variables will change:
int buttonState = 0;  // variable for reading the pushbutton status

void setup() {
  // initialize the LED pins as an output:
  pinMode(ledPinOne, OUTPUT);
  pinMode(ledPinTwo, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}

void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPinOne, HIGH);
    digitalWrite(ledPinTwo, LOW);
  }
  else {
    // turn LED off:
    digitalWrite(ledPinOne, LOW);
    digitalWrite(ledPinTwo, HIGH);
  }
}

3. Verify the project and Upload the program. (Don’t forget to power the board).

4. Pin 13’s LED should be on and Pin 12’s LED off. Press the button, and note that Pin 13’s LED turns off and Pin 12’s LED turns on.
Serial Monitor

The Intel Galileo can output text to a serial monitor on the Arduino IDE for debugging or user interfacing.

We’ll start by adding some code to our last example.

Add “Serial.begin(9600)” in the setup function.

This initializes the Serial communication at a baud rate of 9600. The baud rate is the rate at which it transmits data (9600 is default).

```c
void setup() {
  Serial.begin(9600);
  // initialize the LED pins as an output:
  pinMode(ledPinOne, OUTPUT);
}
```

Next we will attempt some debugging. After reading the button Pin, add “Serial.println(buttonState)” to your code.

```c
void loop() {
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);
}
```

Verify and upload your code. Once it has uploaded and started running, click the Serial monitor icon on the top-right of the Arduino IDE.
You’ll see that as you push the button, the Serial Monitor outputs the value of “buttonState” which is the digitalRead value of the button.

This is great for debugging as you can easily distinguish whether you are experiencing a hardware or software issue.

**Exercises**

**Video Demonstration:** [https://youtu.be/6kj8lNUxqHY](https://youtu.be/6kj8lNUxqHY)

1. A car has a fuel-level sensor that sets A0..A3 to a value between 0 (empty) and 15 (full). A series of LEDs connected to Pin8..Pin13 should light to graphically indicate the fuel level. If the fuel level is 1 or 2, Pin 8 lights. If the level is 3 or 4, Pin 8 and Pin 9 light. 5-6 lights Pin8..Pin10. 7-9 lights Pin8..Pin11. 10-12 lights Pin8..Pin12. 13-15 lights Pin8..Pin13. Also, Pin 7 connects to a "Low fuel" icon, which should light if the level is 4 or less. Use buttons on A0..A3 and mimic the fuel-level sensor with presses.
2. Buttons are connected to A0 and A1. The output for Pin13…Pin8 is initially 0. Pressing A0 increments Pin13…Pin8 (stopping at 9). Pressing A1 decrements Pin13…Pin8 (stopping at 0). If both buttons are depressed (even if not initially simultaneously), Pin13…Pin8 resets to 0. If a reset occurs, both buttons should be fully released before additional increments or decrements are allowed to happen. Use LEDs (and resistors) on Pin13…Pin8. Use a state machine (*not* synchronous) captured in C.

**Note:** Make sure that one button press causes only one increment or decrement respectively. Pressing and holding a button should **NOT** continually increment or decrement the counter.

**Note:** This state machine (increment/decrement) will be reused several times in subsequent lessons.

**Challenge Problem**

3. Create your own festive lights display with 6 LEDS connected to Pin13…Pin8, lighting in some attractive sequence. Pressing the button on A0 changes the lights to the next configuration in the sequence.