Intel Innovation Week

University of Johannesburg (23 June – 27 June 2014)

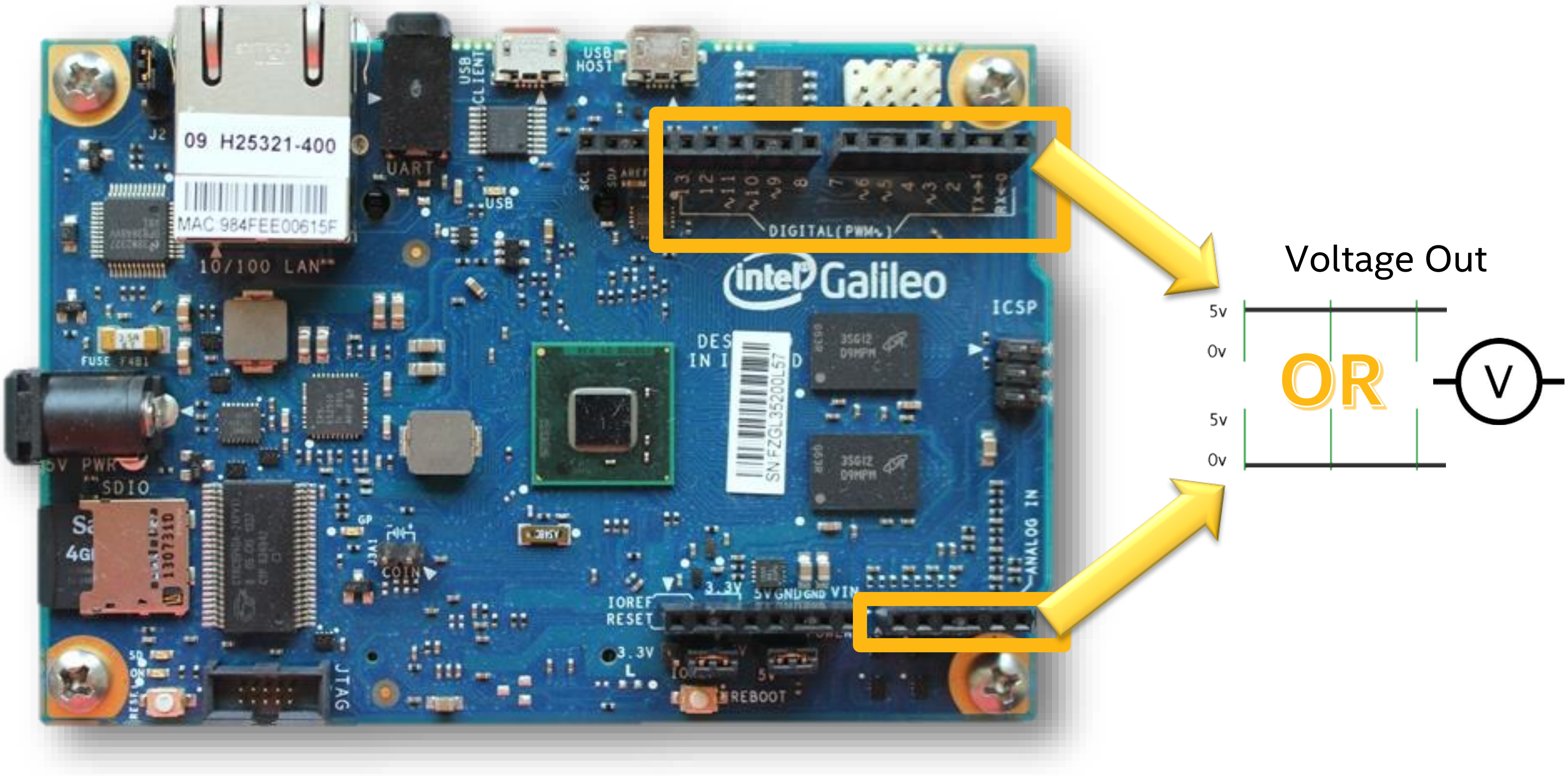
Basic I/O

intel Galileo

What will you make?

Basic I/O - Digital Writes

Digital Write



Voltage Out

5v

0v

OR

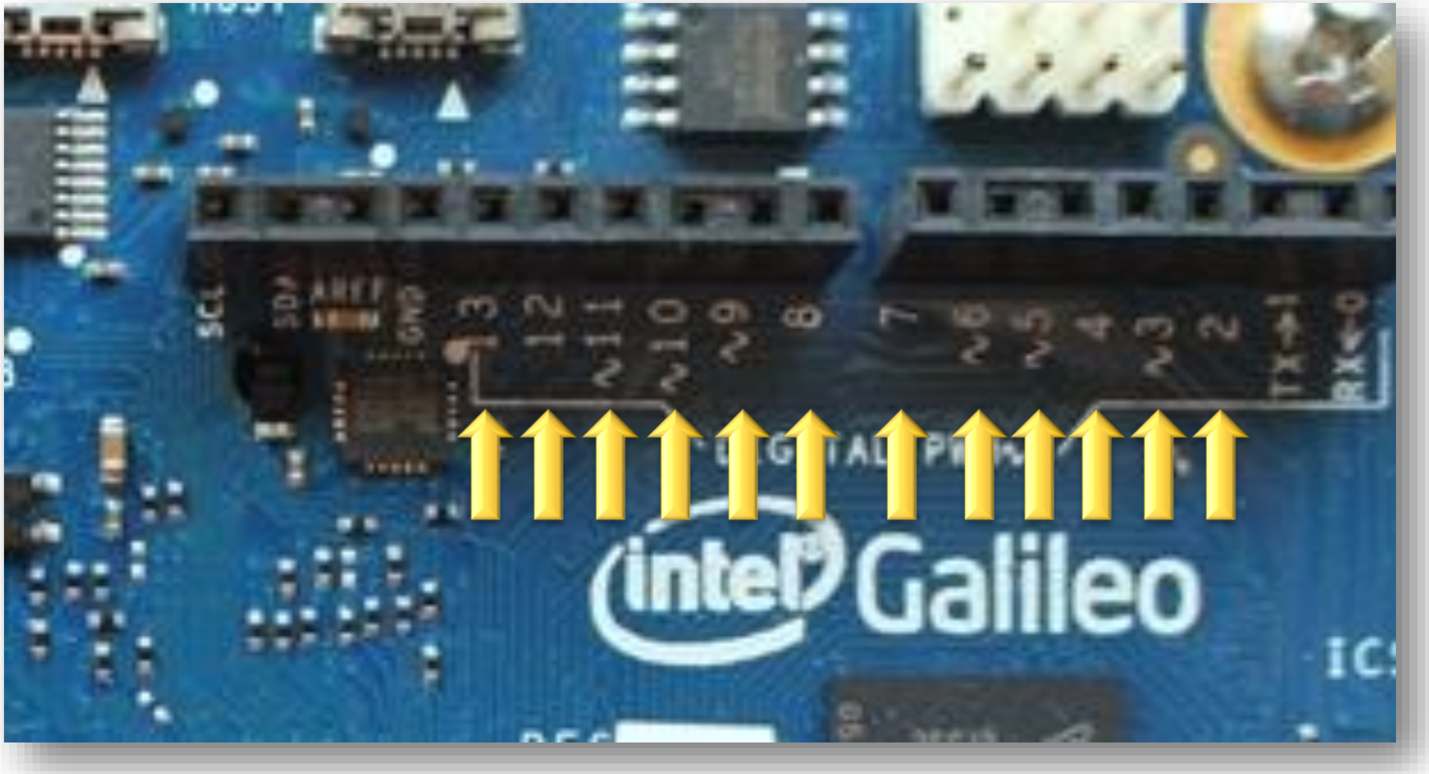
5v

0v

V

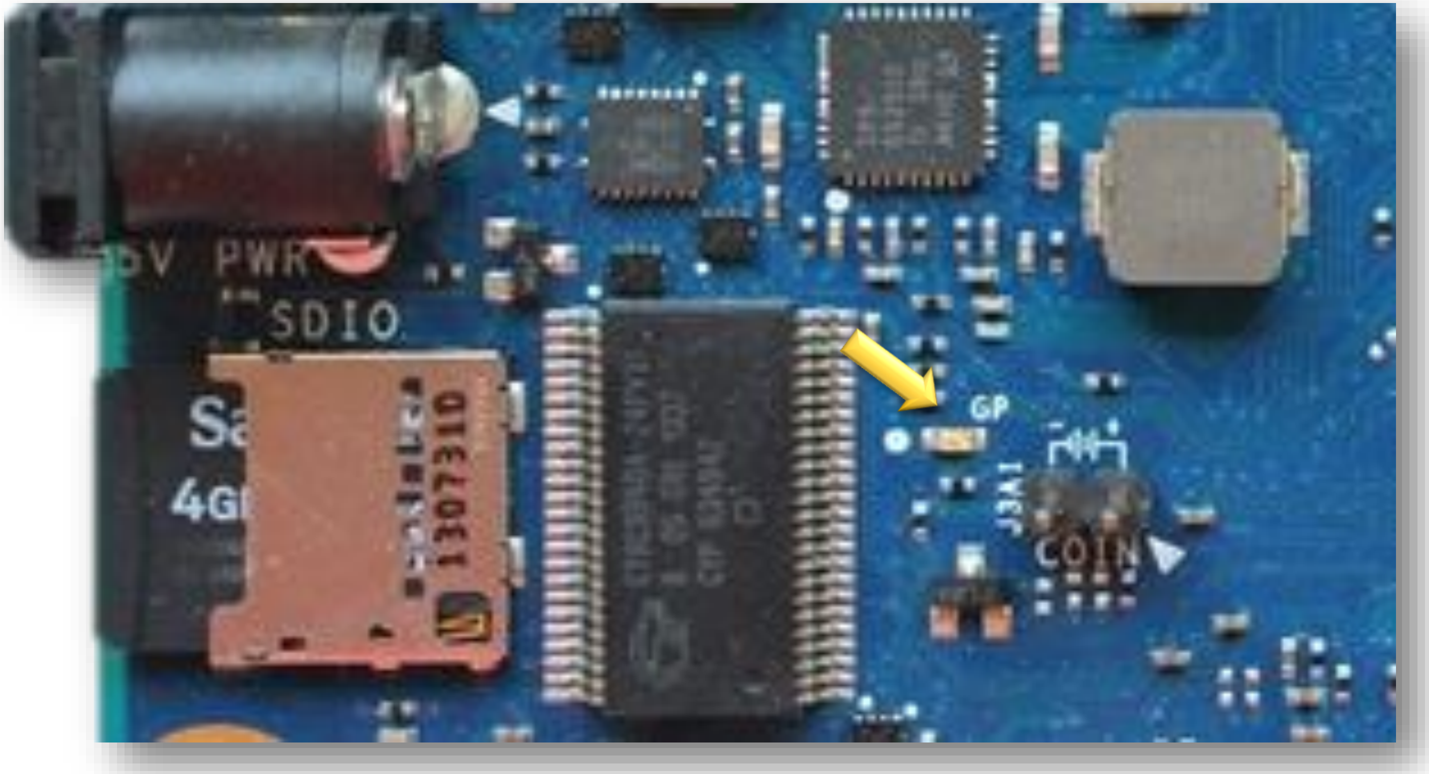
Digital Pins

- General Purpose I/O Pin are 2, 3, 4, 5, 6, 7,8, 9, 10, 11, 12, & 13
- Analog I/O can also be used as digital pings; analog input A0 as digital pin 14 through analog input A5 as digital pin 19



Digital Pins

- Pin 13 has a built-in LED connected to it
- When the pin is HIGH value, the LED is on, when the pin is LOW value, it's off



Digital Write – Key Concepts

pinMode(): Configures the specified pin to behave either as an input or an output

- Syntax: pinMode(pin, mode)
- Example: pinMode(13, OUTPUT) set pin 13 to output mode
- Example: pinMode(13, INPUT) set pin 13 to input mode

digitalWrite(): Write a HIGH or a LOW value to a digital pin

- If set to OUTPUT with pinMode(), 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.
- Syntax: pinWrite(pin, value)
- Example 1: digitalWrite(13, HIGH) is 5 volts to pin 13
- Example 2: digitalWrite(13, LOW) is 0 volts to pin 13

Digital Write – Blink LED

LAB

Project Name: Blink LED *(Instructor Lead)*

Objective: Turns LED on for one second, then off for one second, repeatedly

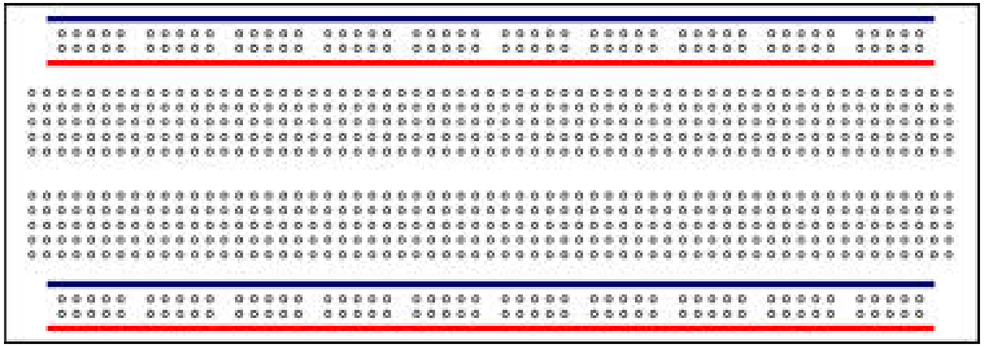
Software Elements

- pinMode()
- digitalWrite()
- delay()

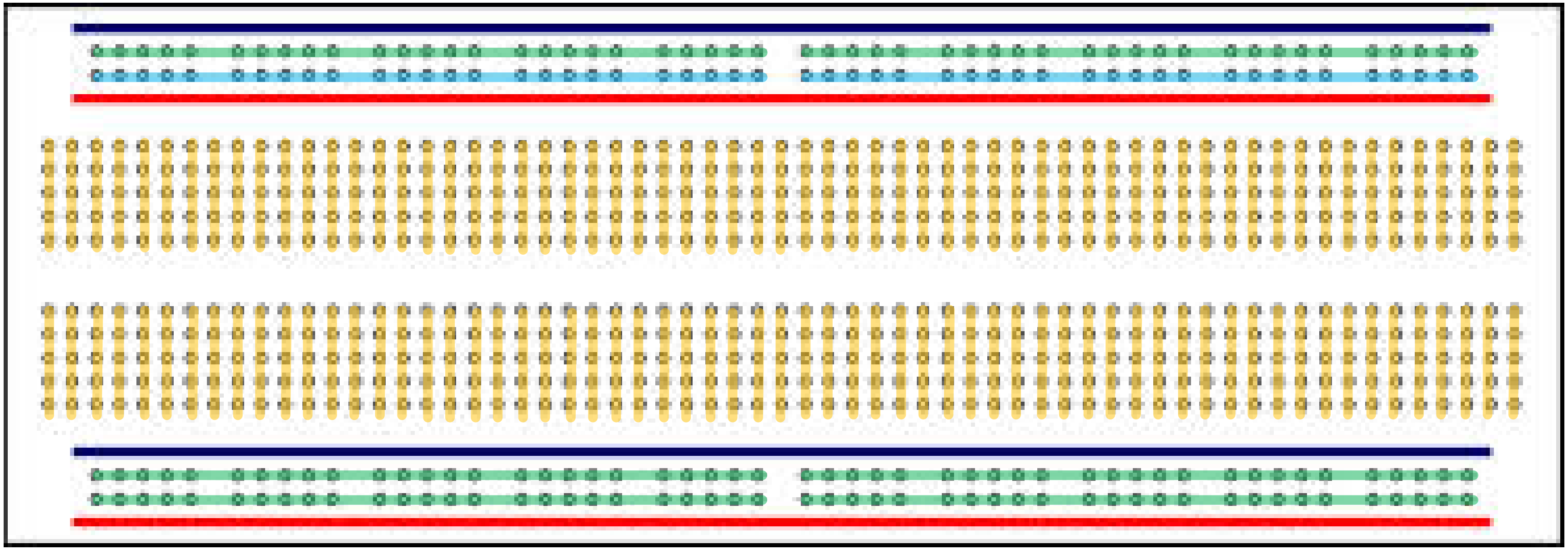
Components

- Intel Galileo Board (Qty 1)
- Breadboard (QTY1)
- Jumper Wires
- 5mm LED (Qty 1)
- 220 Ohm Resister (Qty1)

Breadboard Layout



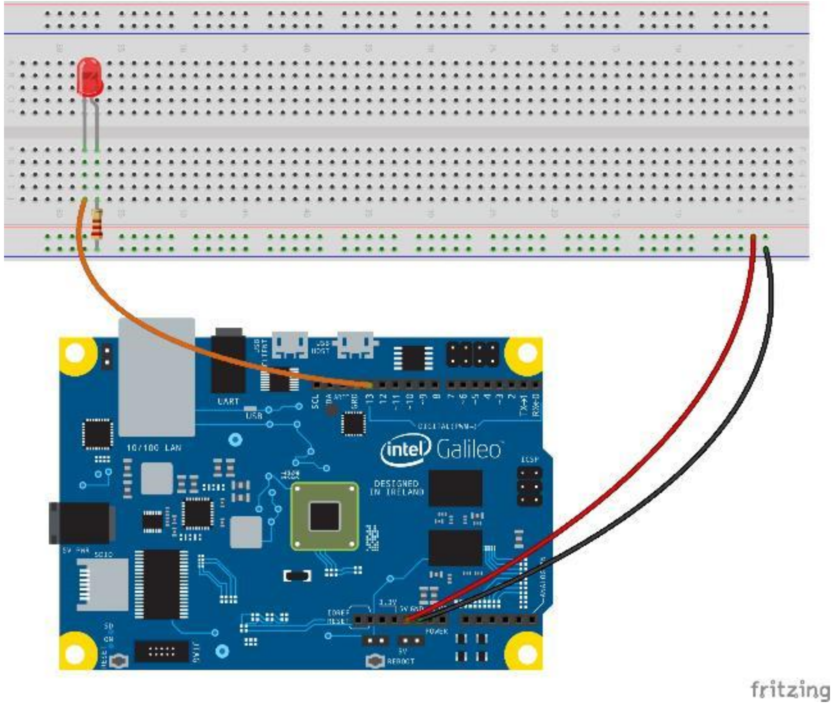
Connected dots
represent connectivity



Digital Write – Blink LED

LAB

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

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 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
}
```

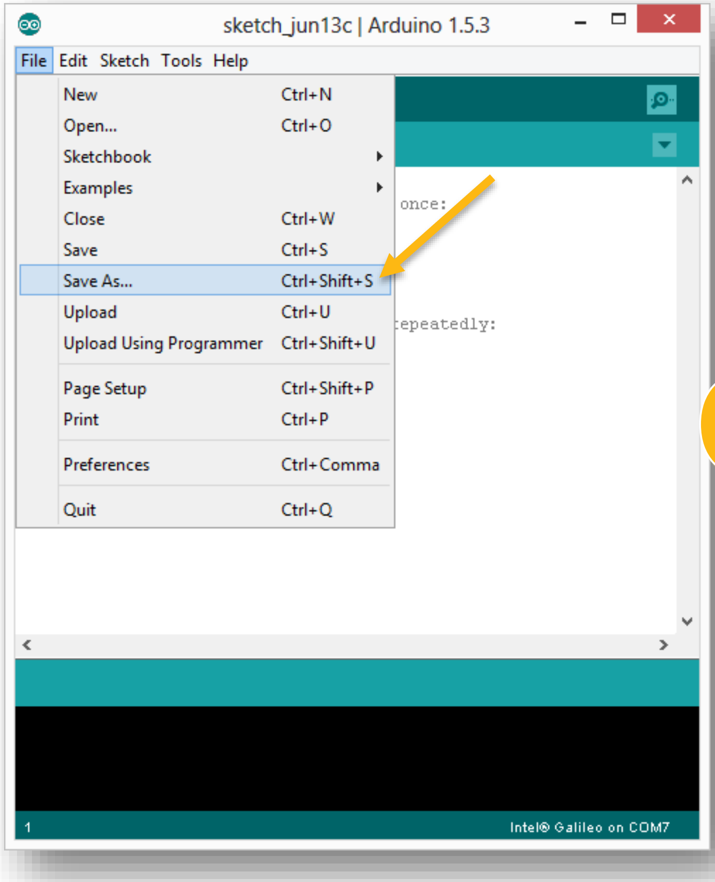


* Other names and brands may be claimed as the property of others

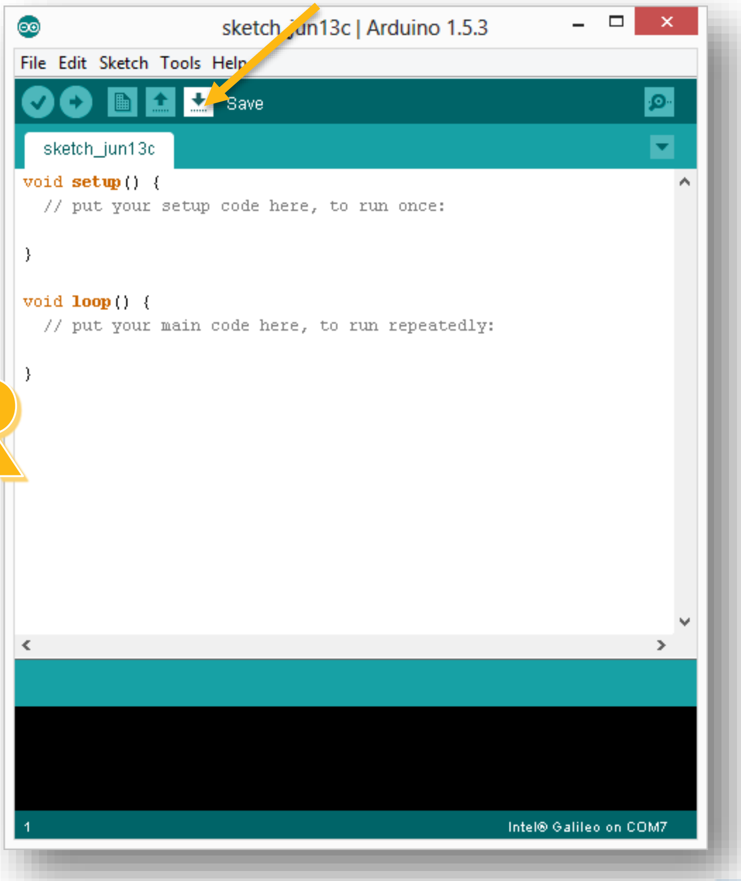
Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section1-DigitalWrite\Blink

Saving a Sketch

1. To Save Sketch, Use “File” -> “Save” or “Save As…”
   Be careful not to over write Sketches you want to keep



OR



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Digital Write - Engineering Challenge LAB

Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

Challenge 1a: Add a Second LED to the circuit and make them Blink together. 1b: Then make them Blink alternating on and off.

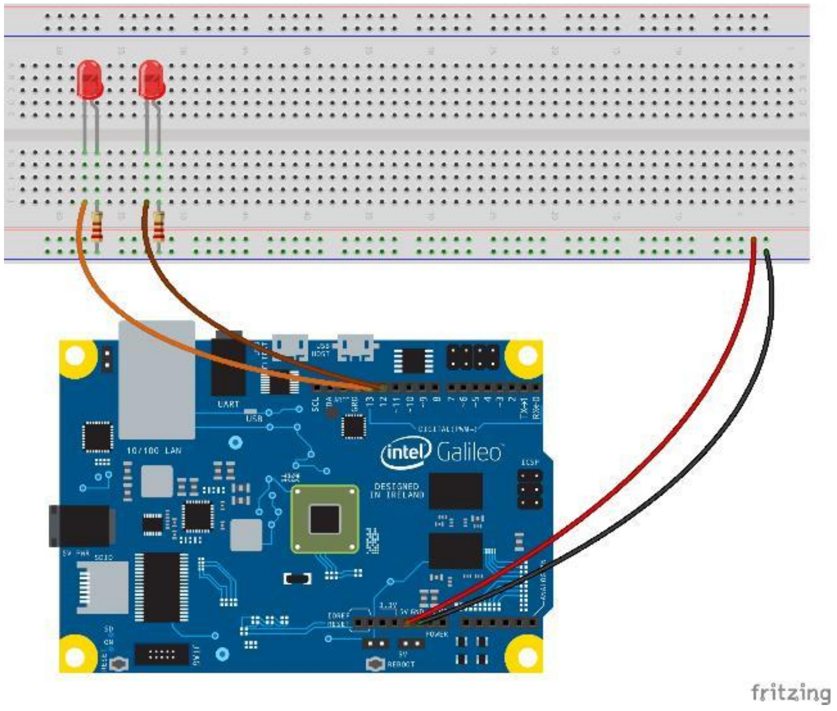
Challenge 2: Add a Third LED to circuit and make them Blink in series and then repeat.

Engineering Challenge Review

Challenge Review

Challenge 1a: Add a Second LED to the circuit and make them Blink together

```
/*
  Blink two LEDs together
  Created by Matt Royer May 5, 2014
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led1 = 13;
int led2 = 12; // Additional LED Pin

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT); // Additional LED Pinmode
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led1, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(led2, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                // wait for a second
  digitalWrite(led1, LOW);    // turn the LED off by making the voltage LOW
  digitalWrite(led2, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                // wait for a second
}
```



* Other names and brands may be claimed as the property of others

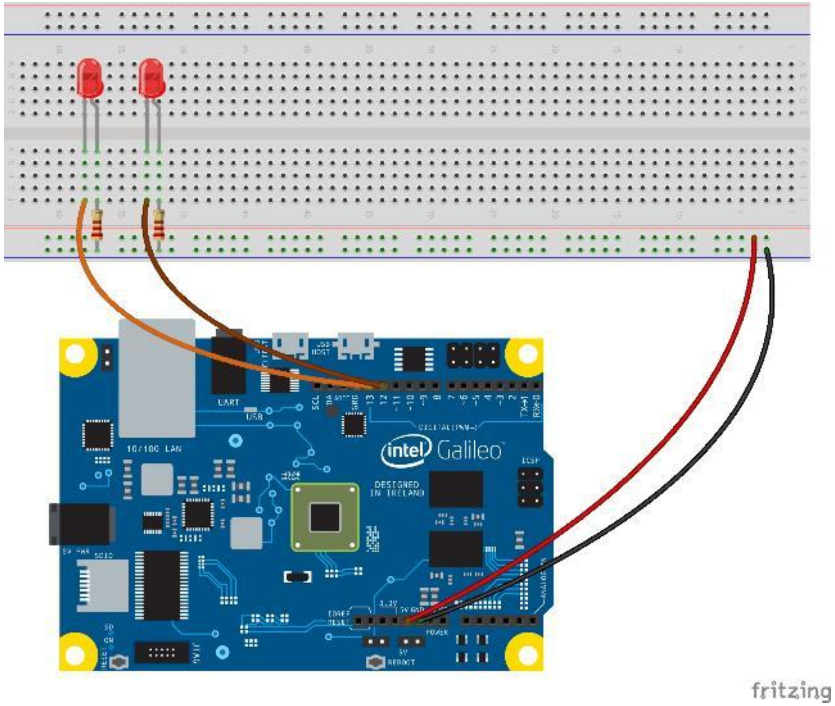
Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section1-DigitalWrite\Blink2Together

Engineering Challenge Review

Challenge Review

Challenge 1b: Make them Blink Alternating on and off.

```
/*
  Blink two LEDs together
  Modified by Matt Royer May 5, 2014
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led1 = 13;
int led2 = 12; // Additional LED Pin

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);  // Additional LED Pinmode
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led1, HIGH);   // turn the LED on (HIGH is the voltage level)
  digitalWrite(led2, LOW);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led1, LOW);    // turn the LED off by making the voltage LOW
  digitalWrite(led2, HIGH);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```



* Other names and brands may be claimed as the property of others

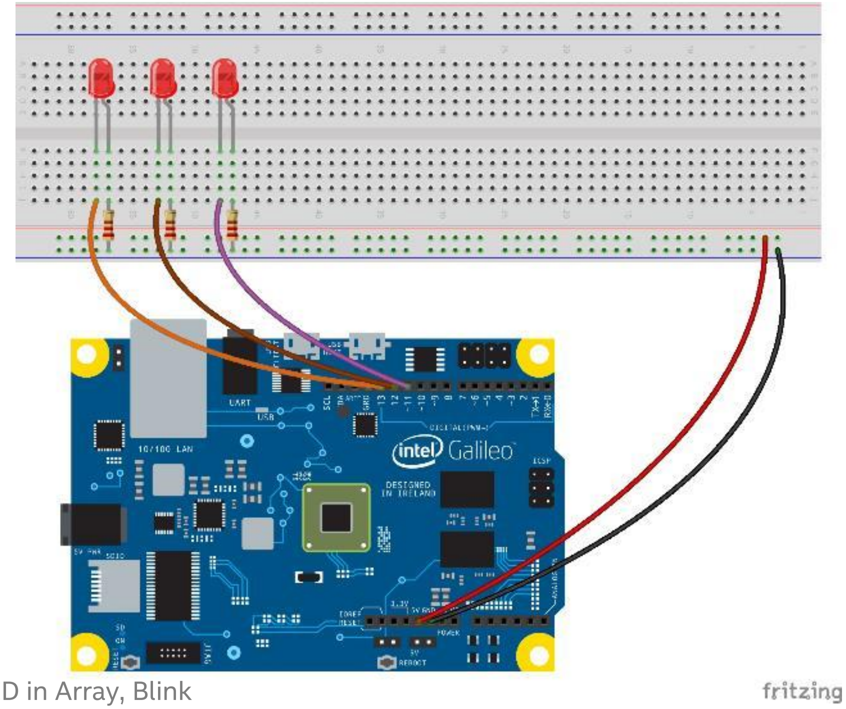
Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section1-DigitalWrite\Blink2Alternating

Engineering Challenge Review

Challenge Review

Challenge 3: Add a Third LED to circuit and make them Blink in series and then repeat.

```
/*
 Blink
 Cycle through Array list of LEDs.  For each LED in Array turns it on for 1 second.

 Modified by Matt Royer May 5, 2014
*/

const int numberOfLED = 3; // Number of LED in Array
const int lEDToBlink[numberOfLED] = { // Array to store LED Pins
  13,
  12,
  11
};

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output
  // For each LED in Array, initialize
  for (int initalizeLED = 0; initalizeLED < numberOfLED; initalizeLED++){
    pinMode(lEDToBlink[initalizeLED], OUTPUT);
  }
}

// the loop routine runs over and over again forever:
void loop() {
  for (int lightLED = 0; lightLED < numberOfLED; lightLED++){ // For each LED in Array, Blink
    digitalWrite(lEDToBlink[lightLED], HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);
    digitalWrite(lEDToBlink[lightLED], LOW);   // turn the LED on (HIGH is the voltage level)
  }
}
```

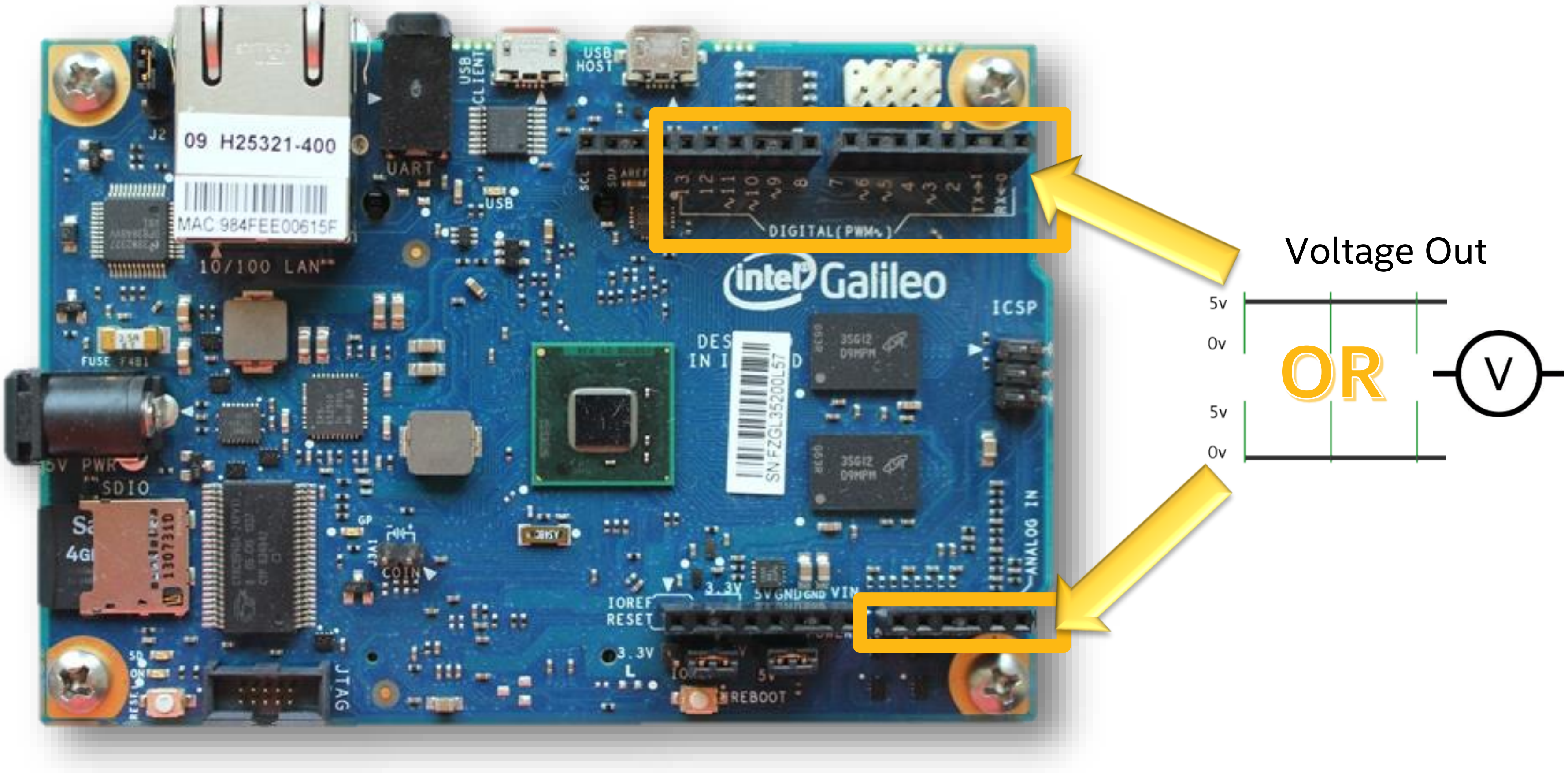


Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section1-DigitalWrite\Blink3Serially

* Other names and brands may be claimed as the property of others

Basic I/O - Digital Reads

Digital Read



Voltage Out

5v

0v

OR

5v

0v

Digital Read – Key Concepts

Reminder pinMode(): Configures the specified pin to behave either as an input or an output

- Syntax: pinMode(pin, mode)
- Example: pinMode(2, OUTPUT) set pin 2 to output mode
- **Example: pinMode(2, INPUT)** set pin 2 to input mode

digitalRead(): Reads the value from a specified digital pin, either HIGH or LOW

- Syntax: pinMode(pin)
- Example 1: digitalRead(2) reads High or Low from pin 2

Digital Read – LED Button Press

LAB

Project Name: LED Button Press*(Instructor Lead)*

Objective: Illuminate LED when button is pressed

Software Elements

- pinMode()
- digitalWrite()
- digitalRead()
- if / else

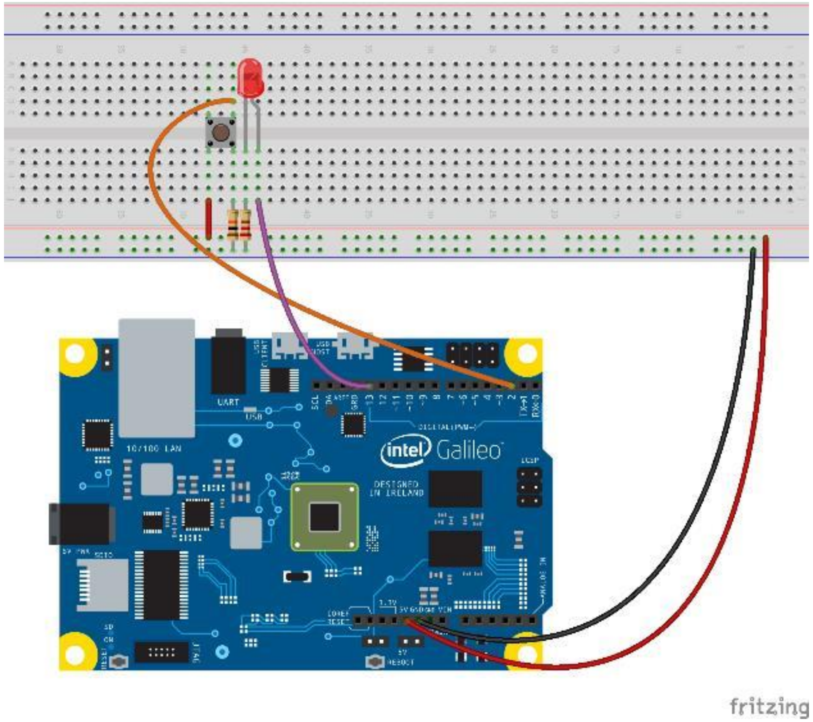
Components

- Intel Galileo Board (Qty 1)
- Breadboard (QTY1)
- Jumper Wires
- 5mm LED (Qty 1)
- Momentary Switch (Qty 1)
- 220 Ohm Resister (Qty1)
- 10k Ohm Resister (Qty1)

Digital Write – LED Button Press

LAB

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;     // the number of the pushbutton pin
const int ledPin =  13;      // the number of the LED pin

// variables will change:
int buttonState = 0;         // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, 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(ledPin, HIGH);
  } else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```



Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section2-DigitalRead\ButtonPush

* Other names and brands may be claimed as the property of others

Digital Write - Engineering Challenge

LAB

Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

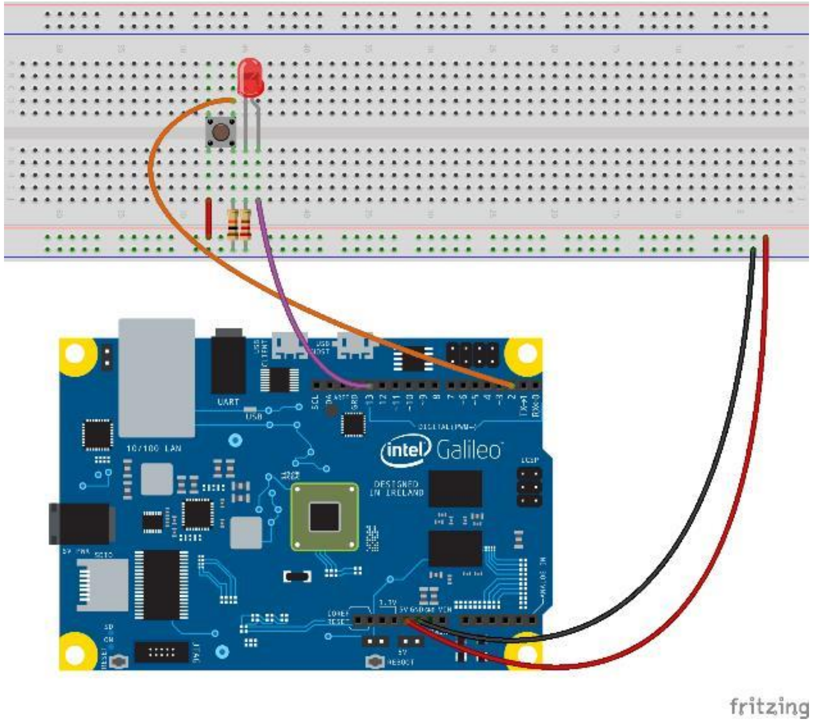
Challenge 1: Blink LED continuously while button is pressed

Engineering Challenge Review

Challenge Review

Challenge 1: Blink LED continuously while button is pressed.

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, 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) {
    digitalWrite(ledPin, HIGH);
    delay(500);
    digitalWrite(ledPin, LOW);
    delay(500);
  }
  digitalWrite(ledPin, LOW);
}
```



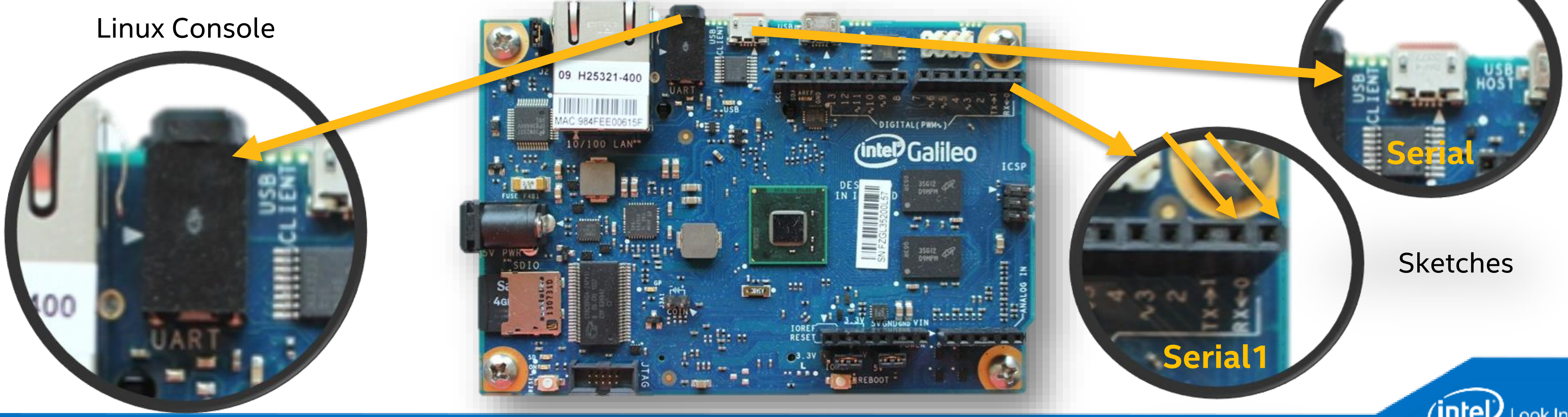
* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section2-DigitalRead\ButtonPushBlink

Basic I/O - Serial Writes

Refresher: Galileo Serial Interfaces

- Used for communication between the Arduino board and a computer or other devices.
- Arduino platforms have at least one serial port (also known as a UART or Universal Asynchronous Receiver/Transmitter)
- Serial Communicates through digital pins 0 (RX) & 1 (TX) and via USB to Computer for Sketches
- Serial Communicates through UART for Linux Console
- Supported baud rates: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, or 115200.



Serial Write – Key Concepts

Serial.begin(): Sets the data rate in bits per second (baud) for serial data transmission

- Syntax: Serial.begin(baud)
- Example: Serial.begin(9600) sets serial baud rate to 9600 bits per second

Serial.print(): Prints data to the serial port as human-readable ASCII text without carriage return / Newline Feed character

- Syntax: Serial.print(val) or Serial.print(val, format)
- Parameters:
  - val: the value to print - any data type
  - format: specifies the number base or number of decimal places
- Example: Serial.print(”Hello world.“) gives "Hello world.“
- Example: Serial.print(1.23456, 2) gives “1.23”

Serial Write – Key Concepts

Serial.println(): Prints data to the serial port as human-readable ASCII text *followed by a carriage return and a newline character*

- Syntax: Serial.println(val) or Serial.print(val, format)
- Parameters:
  - val: the value to print - any data type
  - format: specifies the number base or number of decimal places
- Example: Serial.println(”Hello world.“) gives "Hello world.“
- Example: Serial.println(1.23456, 2) gives “1.23”

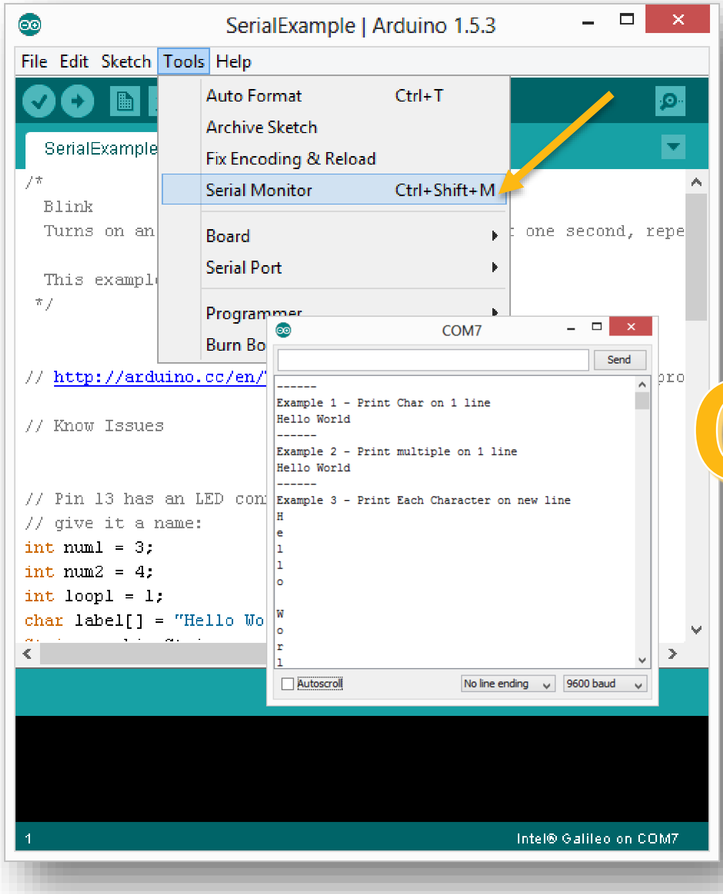
Serial.write(): Writes binary data to the serial port. This data is sent as a byte or series of bytes

- Syntax: Serial.write(val)
- Example: Serial.wrtie(”Hello world“) writes "Hello world“

Bring up Serial Monitor

- Select “Tools” -> “Serial Monitor” or click Serial Monitor Icon

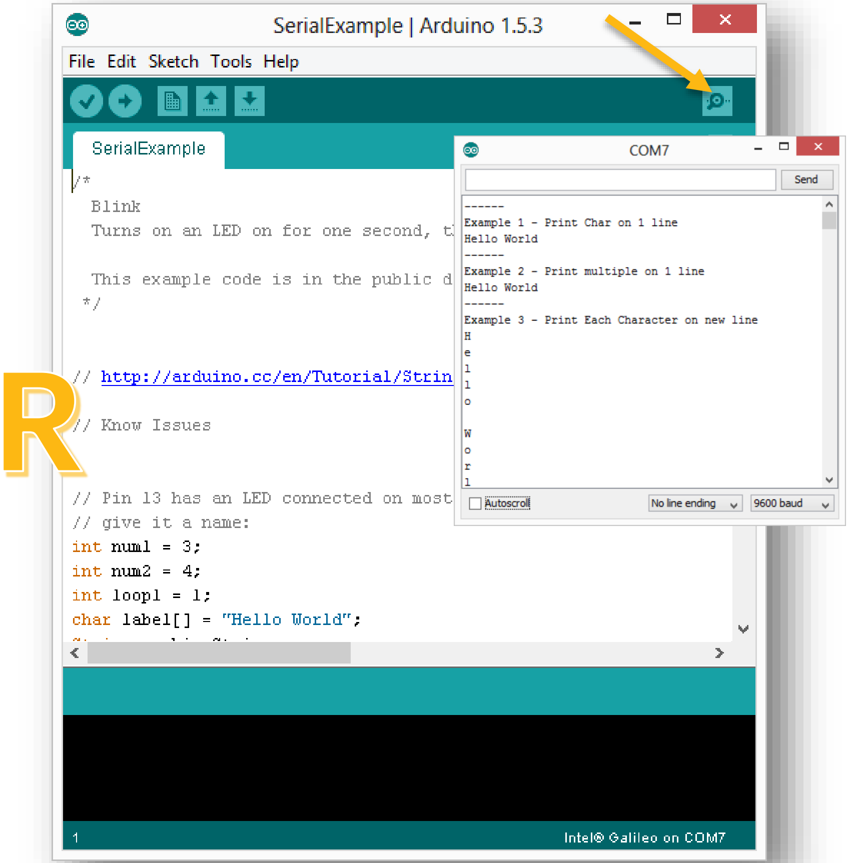
Note: Sketch must be loaded first; else, Serial Monitor will close on Sketch upload



```
/*
  Blink
  Turns on an                              one second, repe

  This exampl
 */


// http://arduino.cc/en/                                pro

// Know Issues


// Pin 13 has an LED con
// give it a name:
int num1 = 3;
int num2 = 4;
int loop1 = 1;
char label[] = "Hello Wo
```

```
------
Example 1 - Print Char on 1 line
Hello World
------
Example 2 - Print multiple on 1 line
Hello World
------
Example 3 - Print Each Character on new line
H
e
l
l
o

W
o
r
l
```

OR



```
/*
  Blink
  Turns on an LED on for one second, t

  This example code is in the public d
 */


// http://arduino.cc/en/Tutorial/Strin

// Know Issues


// Pin 13 has an LED connected on most
// give it a name:
int num1 = 3;
int num2 = 4;
int loop1 = 1;
char label[] = "Hello World";
```

```
------
Example 1 - Print Char on 1 line
Hello World
------
Example 2 - Print multiple on 1 line
Hello World
------
Example 3 - Print Each Character on new line
H
e
l
l
o

W
o
r
l
```

Serial Write – Printing to Serial Monitor LAB

Project Name: Printing to Serial Monitor *(Instructor Lead)*

Objective: Demonstrate different means to print to Serial monitoring

Software Elements

- Serial.Begin()
- Serial.print()
- Serial.println()
- concat()
- delay()

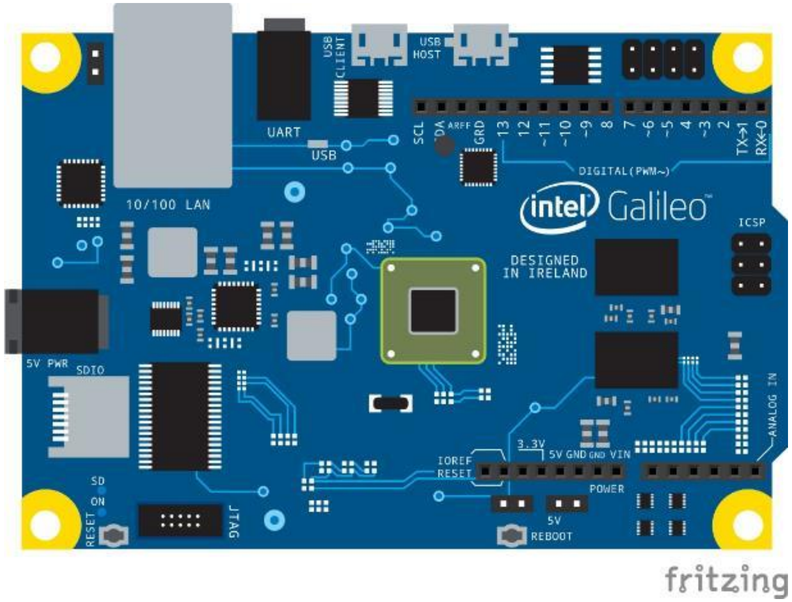
Components

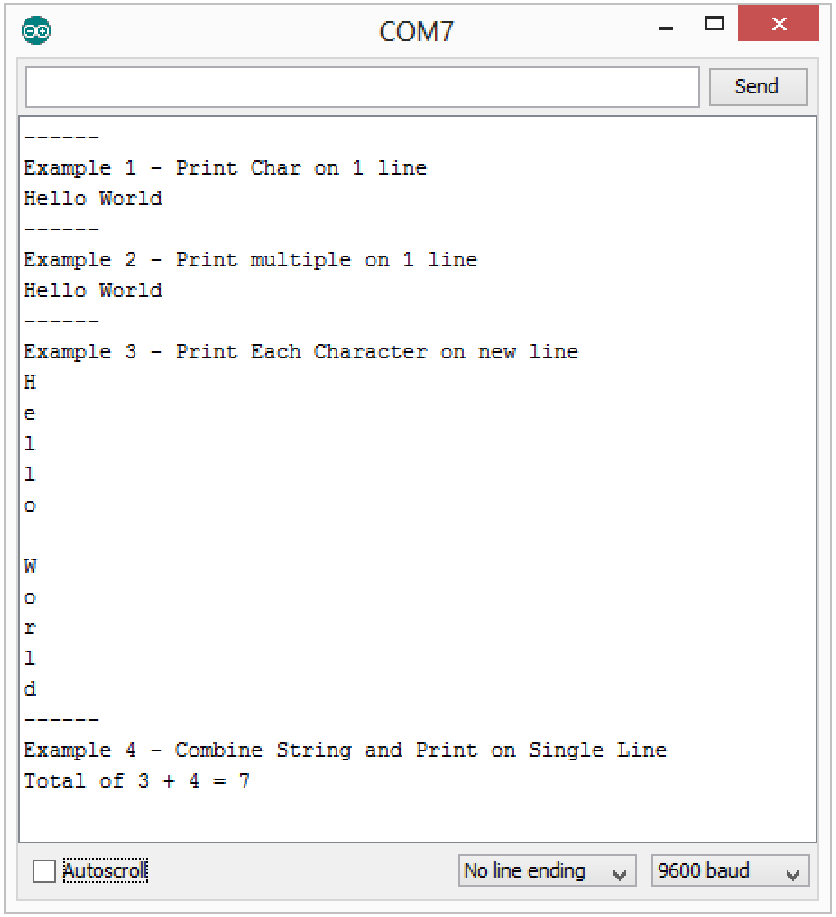
- Intel Galileo Board (Qty 1)

Serial Write – Printing to Serial Monitor

LAB

Source Code: *Lesson2-BasicIO\Section3-SerialWrite\SerialExample*





```
------
Example 1 - Print Char on 1 line
Hello World
------
Example 2 - Print multiple on 1 line
Hello World
------
Example 3 - Print Each Character on new line
H
e
l
l
o

W
o
r
l
d
------
Example 4 - Combine String and Print on Single Line
Total of 3 + 4 = 7
```

□ Autoscroll   No line ending   9600 baud

* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section3-SerialWrite\SerialExample

Serial Write - Engineering Challenge

LAB

Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

Challenge 1: Store your Name in a variable and write it to serial interface

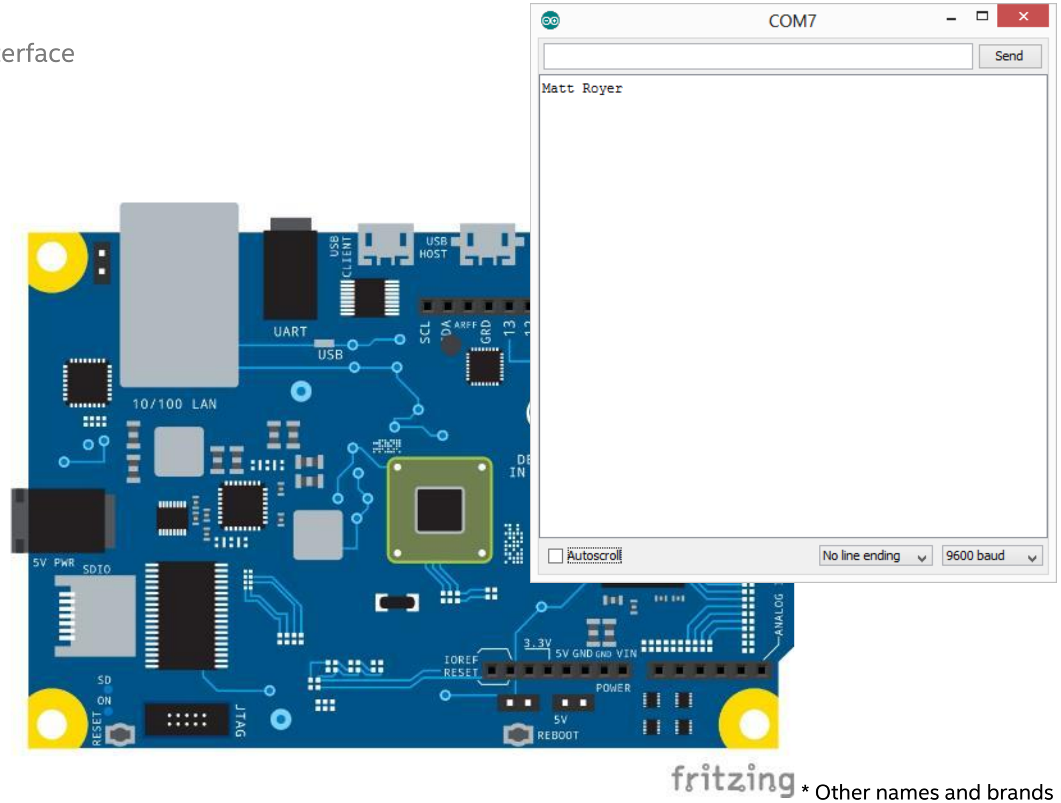
Challenge 2: Write every number from 0-100 to Serial interface

Engineering Challenge Review

Challenge Review

Challenge 1: Store your Name in a variable and write it to serial interface

```
/*
  Store your Name in a variable and write it to serial interface

  Created by Matt Royer
*/

String myName = "Matt Royer";

// the setup routine runs once when you press reset:
void setup() {
  // initialize the Serial Interface
  Serial.begin(9600);

}

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

  Serial.println(myName);

  delay(2000);

}
```



```
Matt Royer
```

fritzing

* Other names and brands may be claimed as the property of others

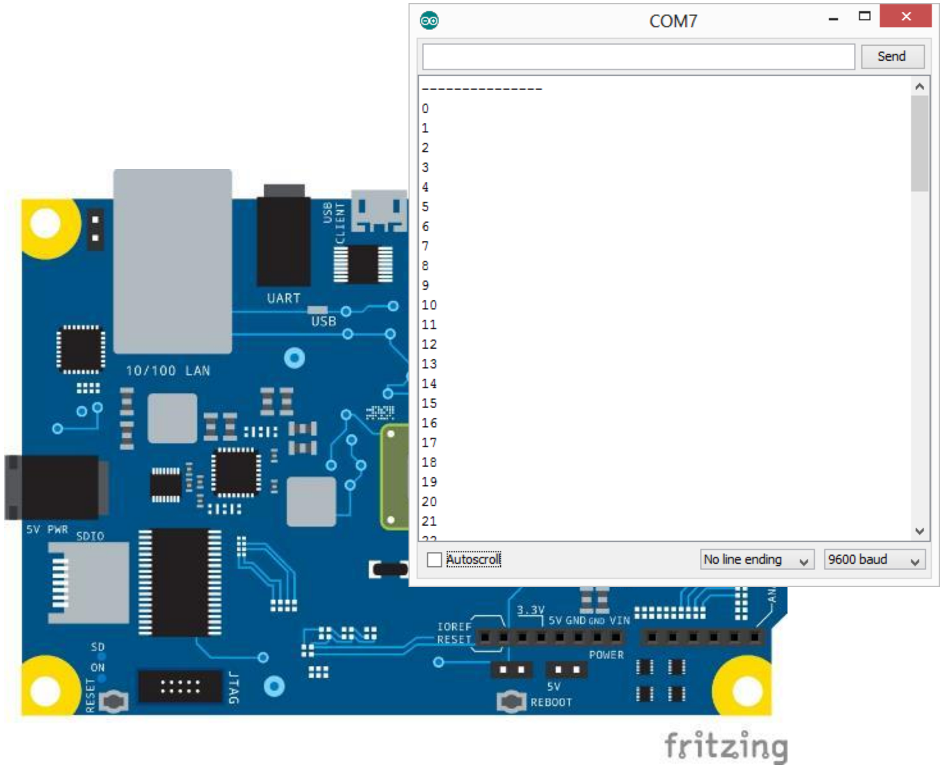
Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section3-SerialWrite\SerialName

Engineering Challenge Review

Challenge Review

Challenge 2: Write every number from 0-100 to Serial interface

```
/*
  Write every number from 0-100 to Serial interface

  Created by Matt Royer
*/

int maxNum = 100;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the Serial Interface
  Serial.begin(9600);

}

// the loop routine runs over and over again forever:
void loop() {
  Serial.println("---------------");
  for (int numIndex = 0; numIndex <=  maxNum; numIndex++){
    Serial.println(numIndex);
  }
  Serial.println("---------------");
  delay(10000);
}
```



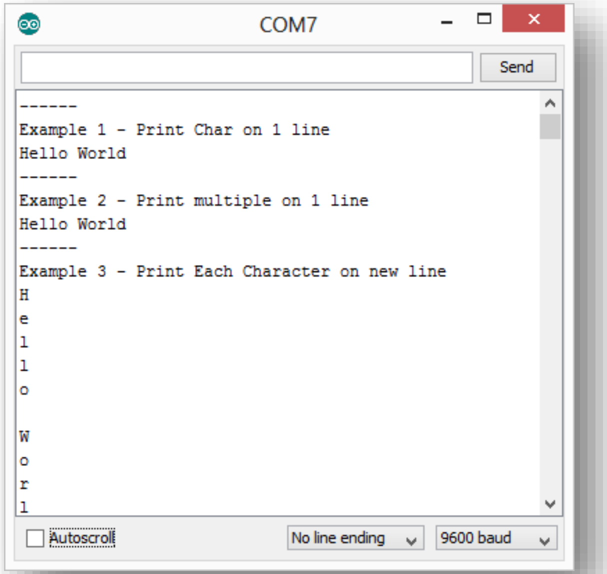
```
---------------
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```

* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section3-SerialWrite\Serial0to100

Using Alternative Terminals

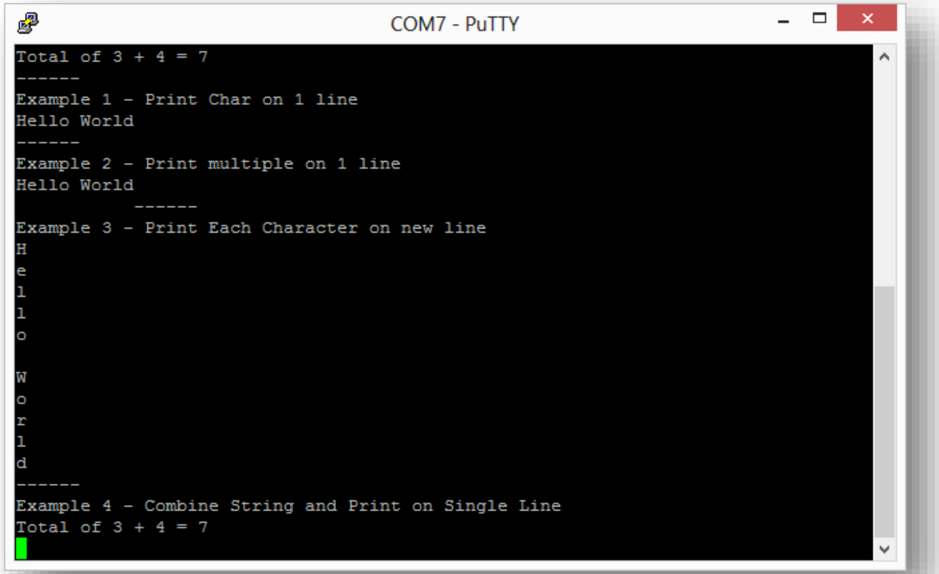
- Although nicely integrated into Arduino IDE, you are not required to use the integrated Serial Monitor.
- Use any Terminal connecting to the proper COM port at the correct baud rate
- Simultaneous use (multiple sessions to COM port), not viable



```
------
Example 1 - Print Char on 1 line
Hello World
------
Example 2 - Print multiple on 1 line
Hello World
------
Example 3 - Print Each Character on new line
H
e
l
l
o

W
o
r
l
```

□ Autoscroll   No line ending   9600 baud

Arduino Serial Monitor



```
Total of 3 + 4 = 7
------
Example 1 - Print Char on 1 line
Hello World
------
Example 2 - Print multiple on 1 line
Hello World
           ------
Example 3 - Print Each Character on new line
H
e
l
l
o

W
o
r
l
d
------
Example 4 - Combine String and Print on Single Line
Total of 3 + 4 = 7
```

PuTTY

Basic I/O - Serial Reads

Serial Read – Key Concepts

Reminder Serial.begin(): Sets the data rate in bits per second (baud) for serial data transmission

- Syntax: Serial.begin(baud)
- Example: Serial.begin(9600) sets serial baud rate to 9600 bits per second

Serial.availible(): Get the number of bytes that have already arrived and been stored in the serial receive buffer (which holds 64 bytes).

- Syntax: Serial.availible()

Serial.read(): Reads incoming serial data. Removes data from serial buffer

- Syntax: Serial.read()

Serial Read – Read Input

LAB

Project Name: Read Input *(Instructor Lead)*

Objective: Read input from Serial Monitor input and write it back out

Software Elements

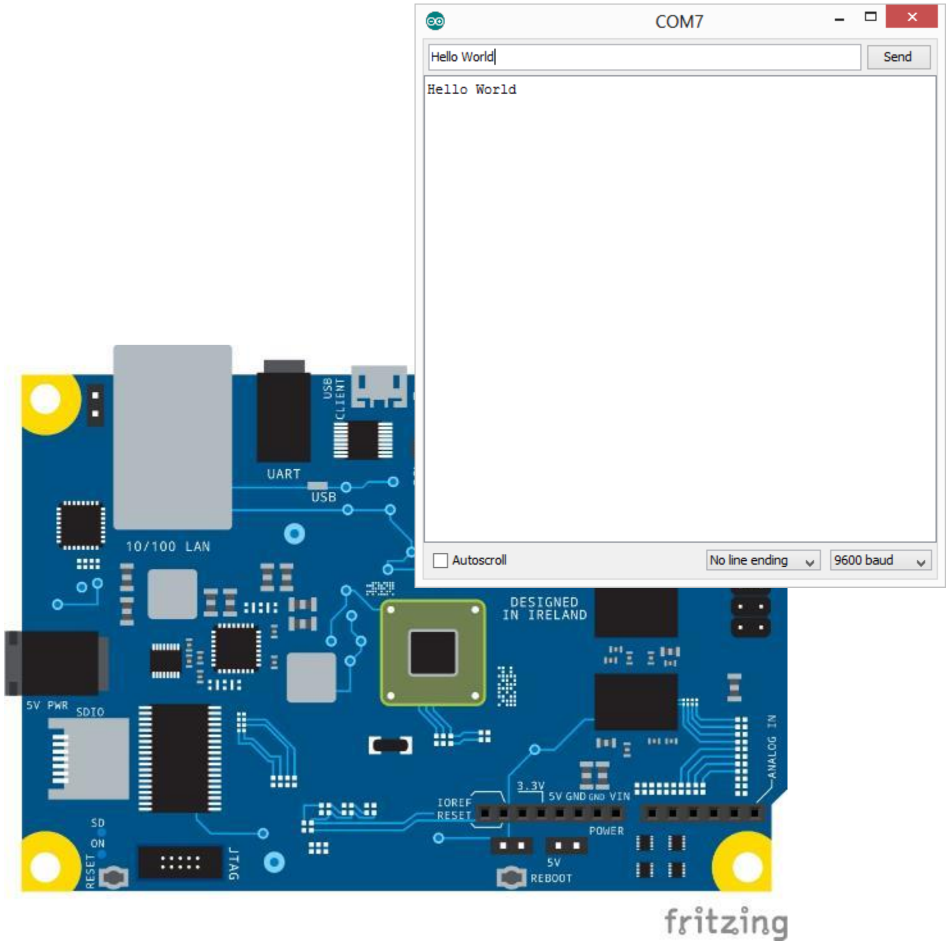
- Serial.begin()
- Serial.availible()
- Serial.read()
- Serial.print()
- Serial.write()
- Serial.println()

Components

- Intel Galileo Board (Qty 1)

Serial Read– Read Input

```
/* Simple Serial ECHO script : Written by ScottC 03/07/2012 */

/* Use a variable called byteRead to temporarily store
   the data coming from the computer */
byte incomingByte;

void setup() {
// Turn the Serial Protocol ON
  Serial.begin(9600);
}

void loop() {
   /*  check if data has been sent from the computer: */
  if (Serial.available()) {
    /* read the most recent byte */
    incomingByte = Serial.read();
    /*ECHO the value that was read, back to the serial port. */
    Serial.write(incomingByte);
  }
}
```



COM7

Hello World | Send

```
Hello World
```

□ Autoscroll | No line ending | 9600 baud

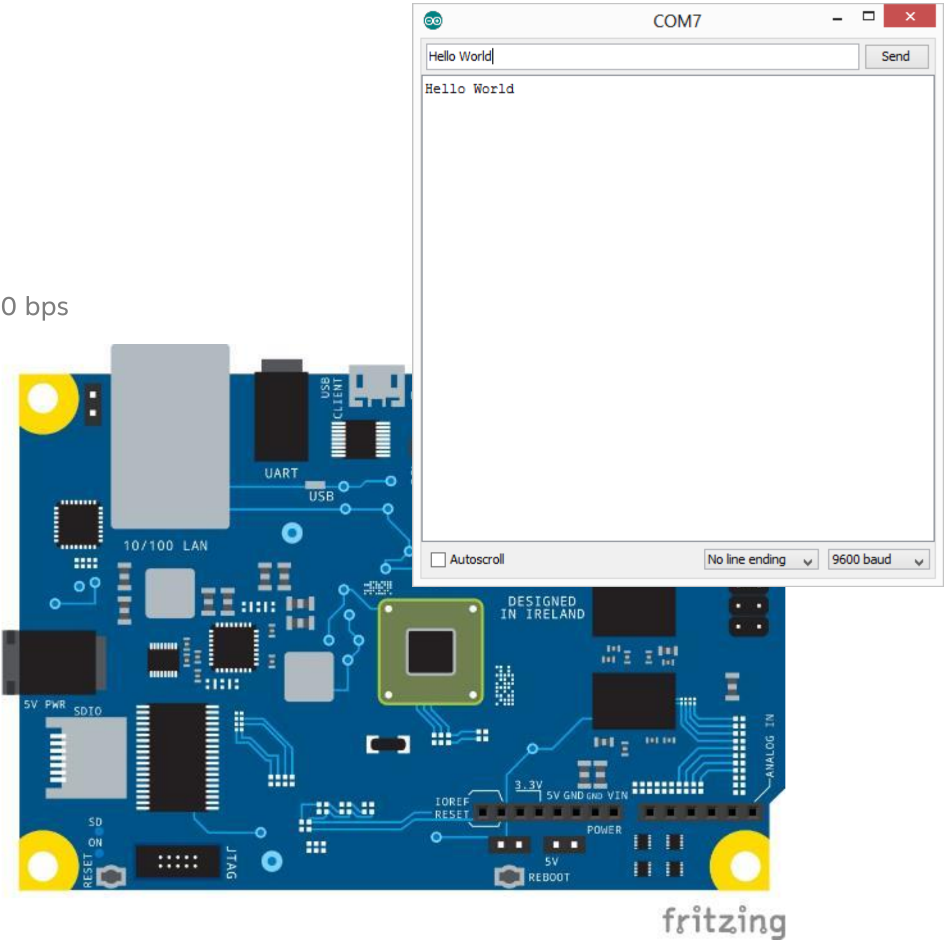
* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section4-SerialRead\SerialReadWrite

Serial Read– Read Entire Buffer

LAB

```
/*
 Read entire input from Serial Monitor input buffer, then write it back out

 Create by Matt Royer
 */


char character;
String content = "";

void setup() {
 Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
}

void loop() {

 while(Serial.available()) {

   character = Serial.read();
   content.concat(character); //Concatenate to existing string

 }

 if (content != "") {

   Serial.println(content); //Print string to serial
   content = ""; //Empty string

 }

}
```



COM7

Hello World

Send

```
Hello World
```

□ Autoscroll

No line ending

9600 baud

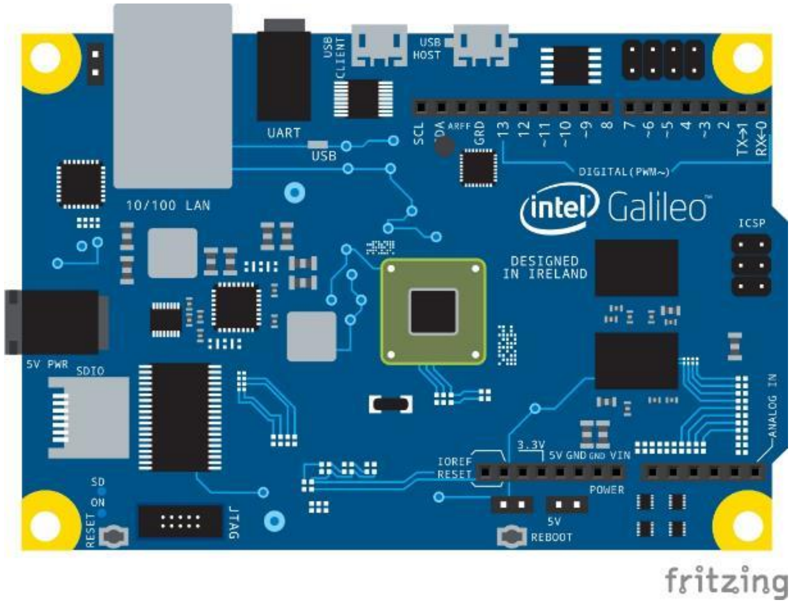
* Other names and brands may be claimed as the property of others

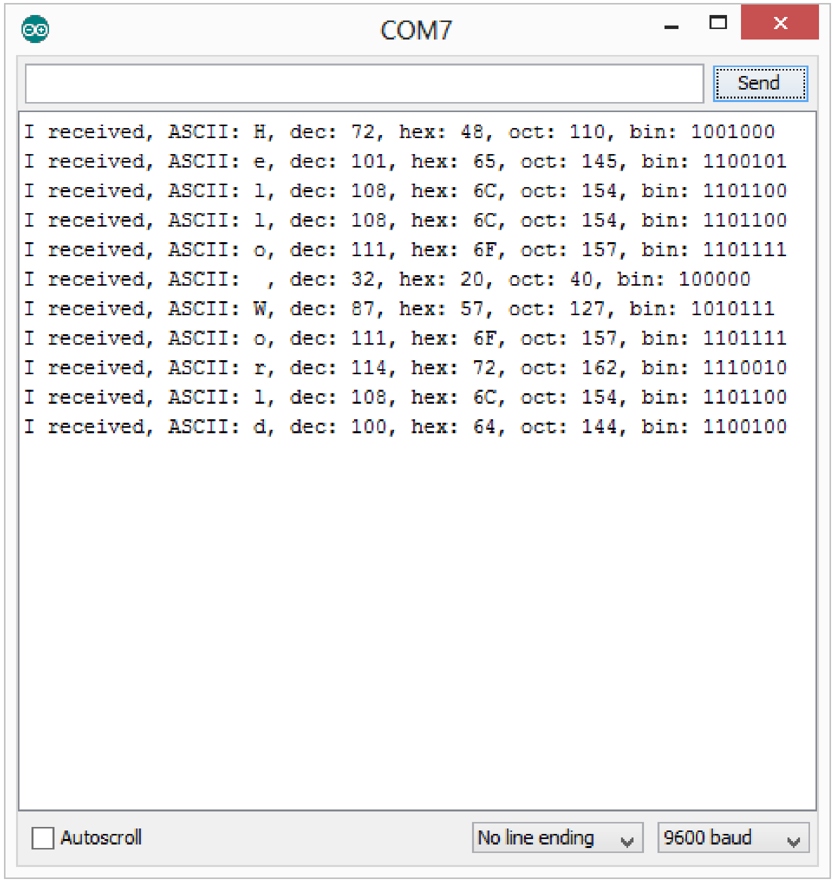
Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section4-SerialRead\Serial_ReadEntireBuffer

Serial Write – Read Input and Format

LAB

Source Code: *Lessons\Lesson2-BasicIO\Section4-SerialRead\Serial_ReadWriteFormates*





```
I received, ASCII: H, dec: 72, hex: 48, oct: 110, bin: 1001000
I received, ASCII: e, dec: 101, hex: 65, oct: 145, bin: 1100101
I received, ASCII: l, dec: 108, hex: 6C, oct: 154, bin: 1101100
I received, ASCII: l, dec: 108, hex: 6C, oct: 154, bin: 1101100
I received, ASCII: o, dec: 111, hex: 6F, oct: 157, bin: 1101111
I received, ASCII:  , dec: 32, hex: 20, oct: 40, bin: 100000
I received, ASCII: W, dec: 87, hex: 57, oct: 127, bin: 1010111
I received, ASCII: o, dec: 111, hex: 6F, oct: 157, bin: 1101111
I received, ASCII: r, dec: 114, hex: 72, oct: 162, bin: 1110010
I received, ASCII: l, dec: 108, hex: 6C, oct: 154, bin: 1101100
I received, ASCII: d, dec: 100, hex: 64, oct: 144, bin: 1100100
```

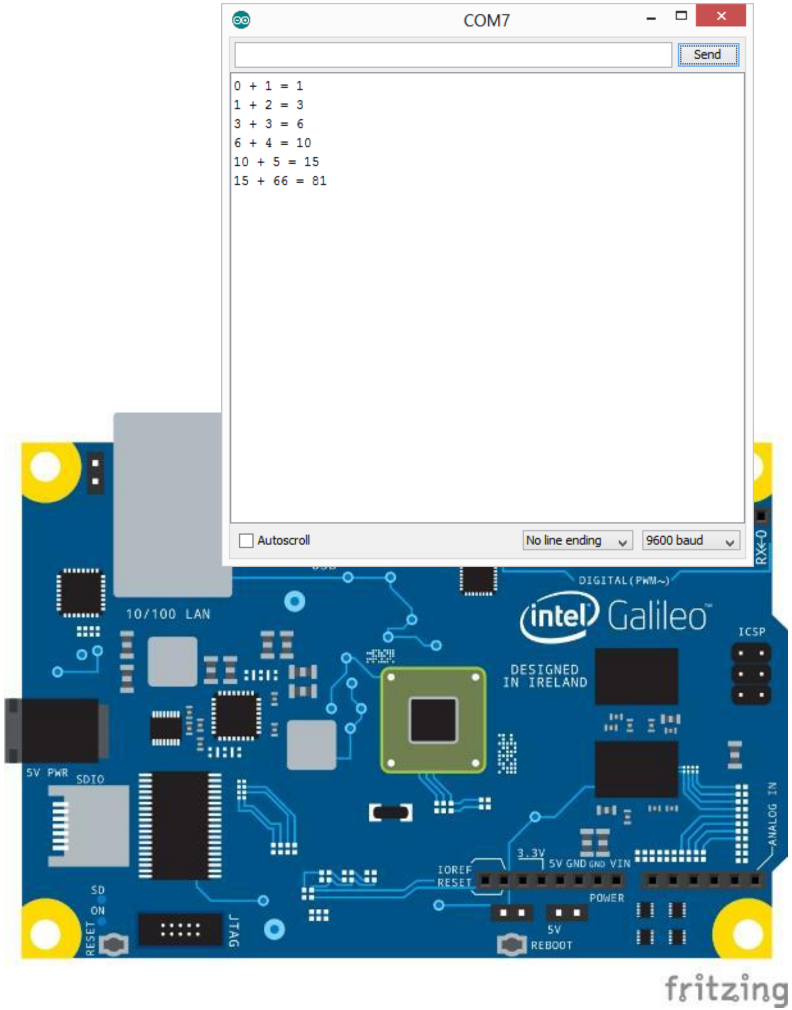
☐ Autoscroll | No line ending | 9600 baud

* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section4-SerialRead\Serial_ReadWriteFormates

Serial Read – Read Convert to int, then Add

```
char character;
String content = "";
int additionSum = 0;


void setup() {
 Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps

}

void loop() {

 //Read Input Buffer
 while(Serial.available()) {
   character = Serial.read();
   content.concat(character);
 }

 if (content != "") {
   if (content.toInt() > 0) { //Make sure input is a number greater than zero

     Serial.print(additionSum);
     Serial.print(" + ");
     Serial.print(content);
     Serial.print(" = ");
     additionSum = additionSum + content.toInt(); // Perform Addition
     Serial.println(additionSum);

   } else {

     Serial.println("Input my be a number greater than 0");
   }
   content = ""; //Empty string
 }
}
```



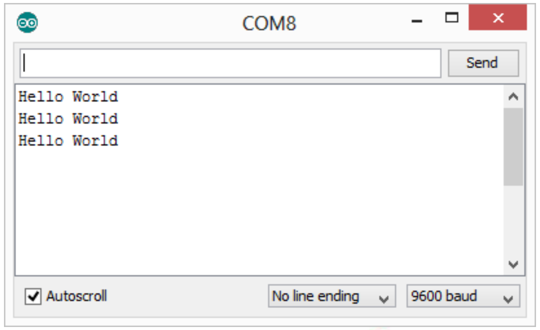
```
0 + 1 = 1
1 + 2 = 3
3 + 3 = 6
6 + 4 = 10
10 + 5 = 15
15 + 66 = 81
```

* Other names and brands may be claimed as the property of others

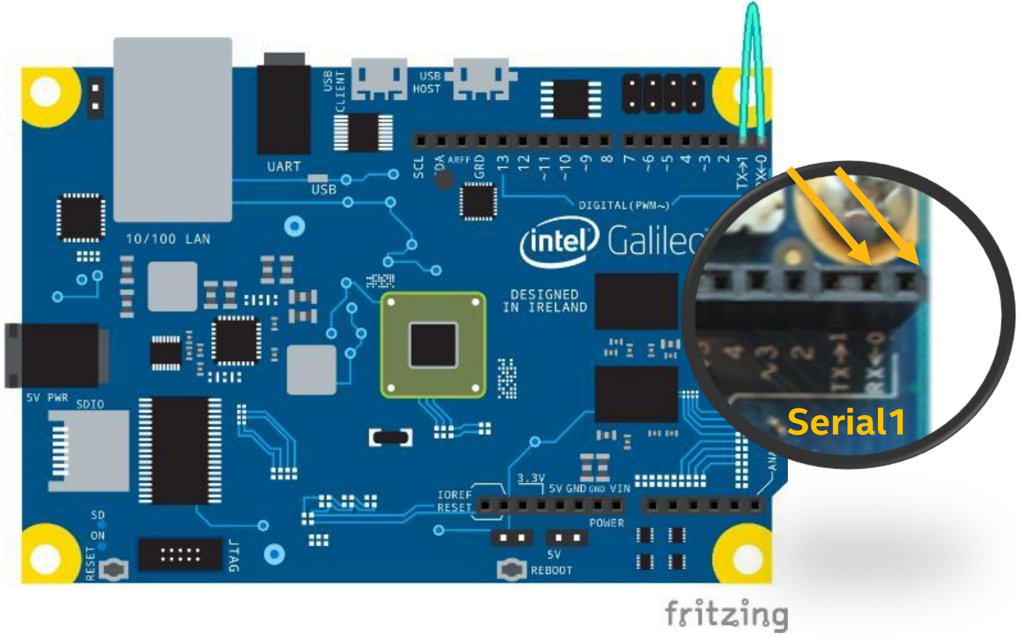
Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section4-SerialRead\Serial_ReadAdd

Serial1 – Reading and Writing

```
char character;
String content = "";

void setup() {
  Serial.begin(115200);     // opens serial port, sets data rate to 115200 bps
  Serial1.begin(115200);     // opens serial port, sets data rate to 115200 bps
}

void loop() {

  Serial1.write("Hello World"); // Write content to Serial1 interface

  while(Serial1.available()) {

    character = Serial1.read();
    content.concat(character); //Concatenate to existing string

  }

  if (content != "") {

    Serial.println(content); //Print string to serial
    content = ""; //Empty string

  }

  delay(2000);

}
```



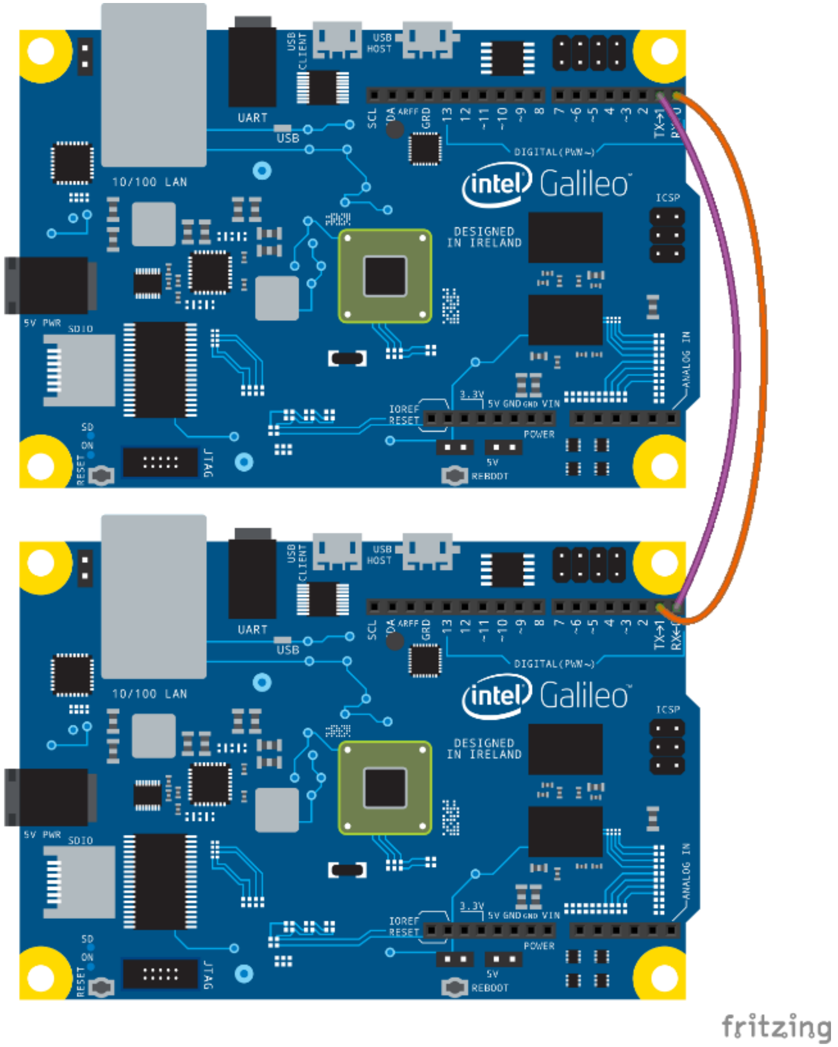
```
Hello World
Hello World
Hello World
```

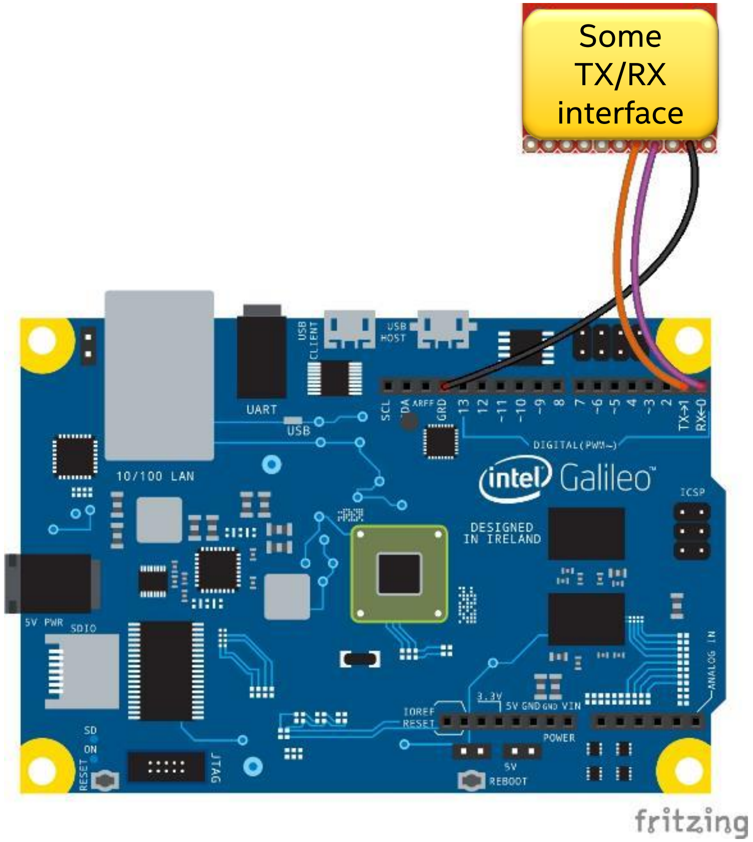


* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section4-SerialRead\Serial1ToSerial

Other Serial1 Possibilities





* Other names and brands may be claimed as the property of others

Serial Write - Engineering Challenge

LAB

Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

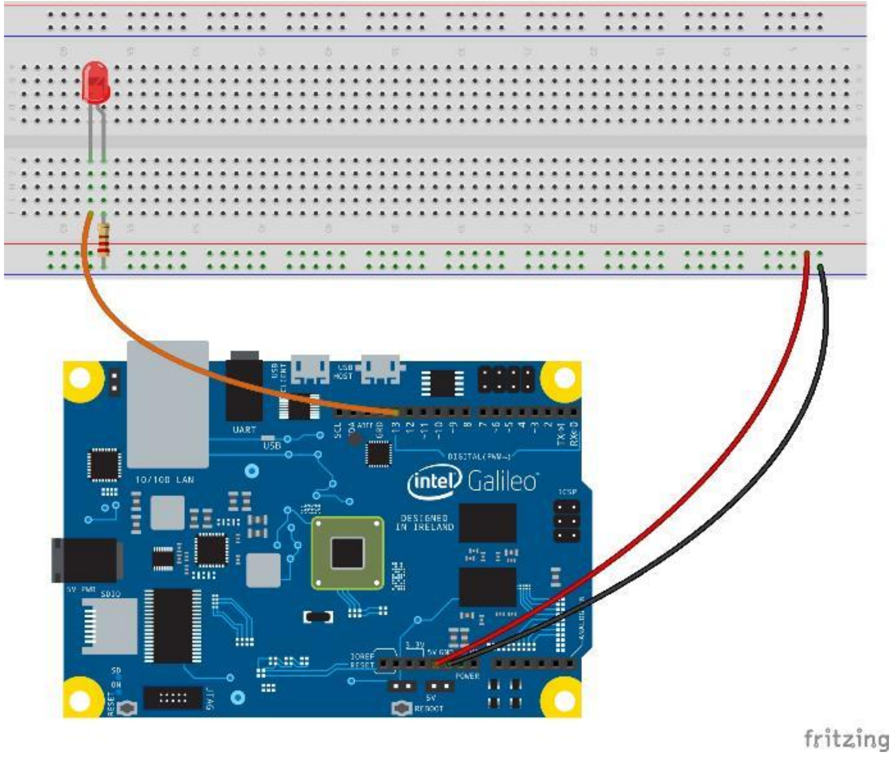
Challenge 1: Type in the number of times to blink an LED

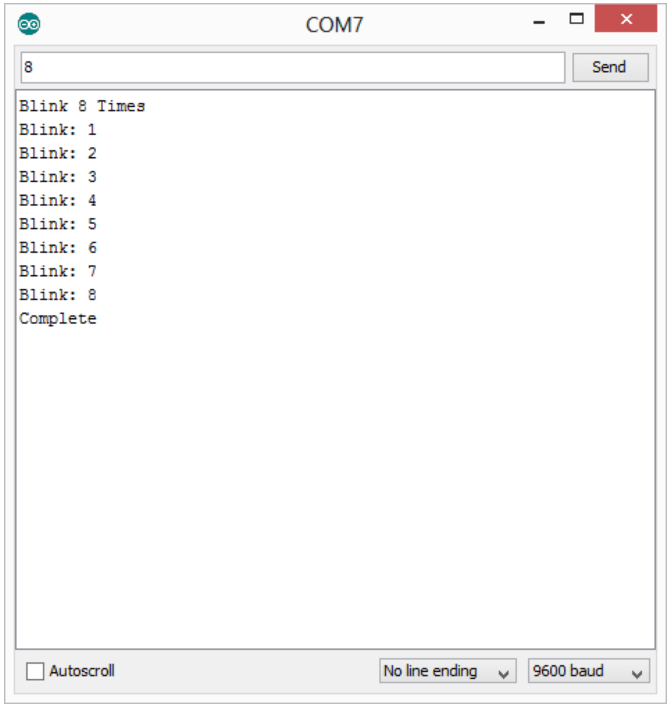
Engineering Challenge Review

Challenge Review

Challenge 1: Type in the number of times to blink an LED

Source Code: *Lessons\Lesson2-BasicIO\Section4-SerialRead\Serial_ReadBlink*





8

```
Blink 8 Times
Blink: 1
Blink: 2
Blink: 3
Blink: 4
Blink: 5
Blink: 6
Blink: 7
Blink: 8
Complete
```

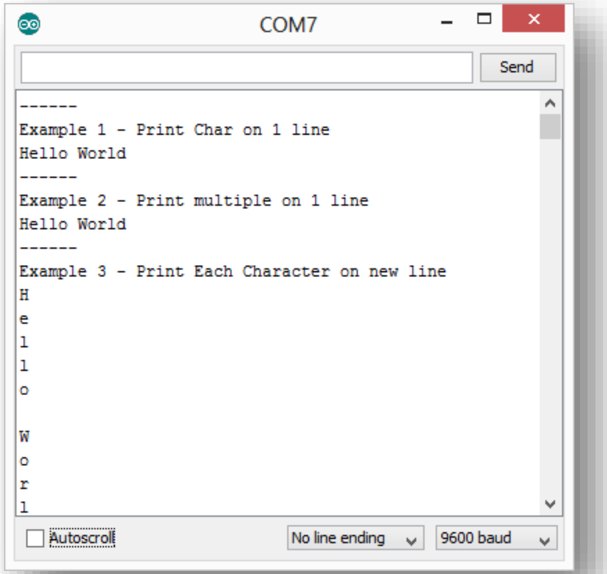
☐ Autoscroll   No line ending   9600 baud

* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section4-SerialRead\Serial_ReadBlink

Using Alternative Terminals

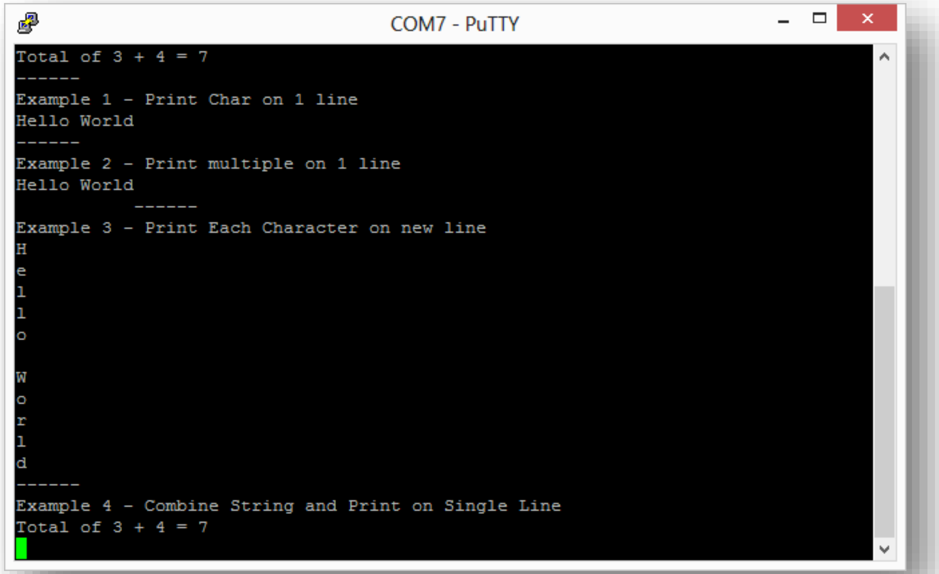
- Although nicely integrated into Arduino IDE, you are not required to use the integrated Serial Monitor.
- Use any Terminal connecting to the proper COM port at the correct baud rate
- Simultaneous use (multiple sessions to COM port), not viable



```
------
Example 1 - Print Char on 1 line
Hello World
------
Example 2 - Print multiple on 1 line
Hello World
------
Example 3 - Print Each Character on new line
H
e
l
l
o

W
o
r
l
```

□ Autoscroll   No line ending   9600 baud

Arduino Serial Monitor

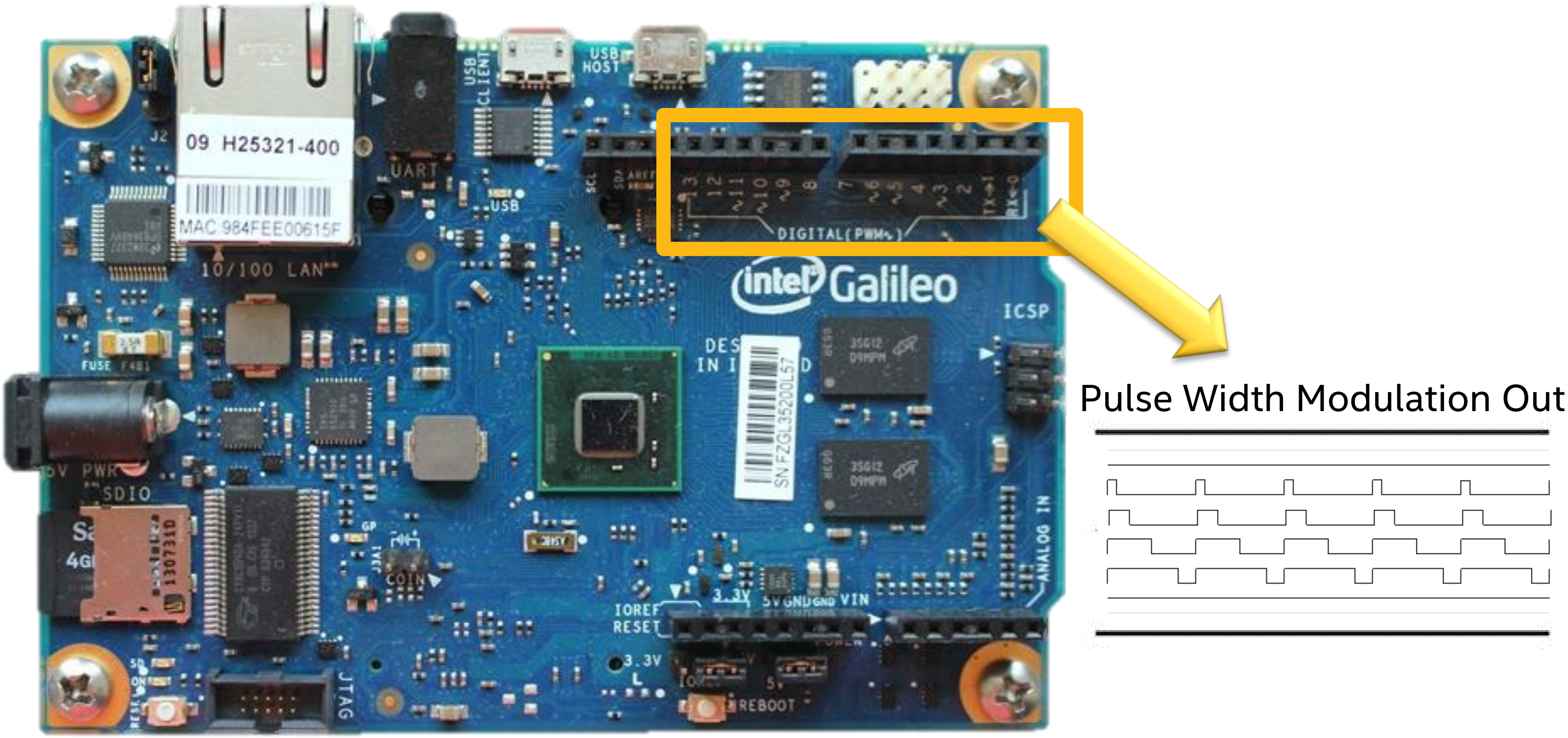


```
Total of 3 + 4 = 7
------
Example 1 - Print Char on 1 line
Hello World
------
Example 2 - Print multiple on 1 line
Hello World
           ------
Example 3 - Print Each Character on new line
H
e
l
l
o

W
o
r
l
d
------
Example 4 - Combine String and Print on Single Line
Total of 3 + 4 = 7
```

PuTTY

Basic I/O - Analog Writes

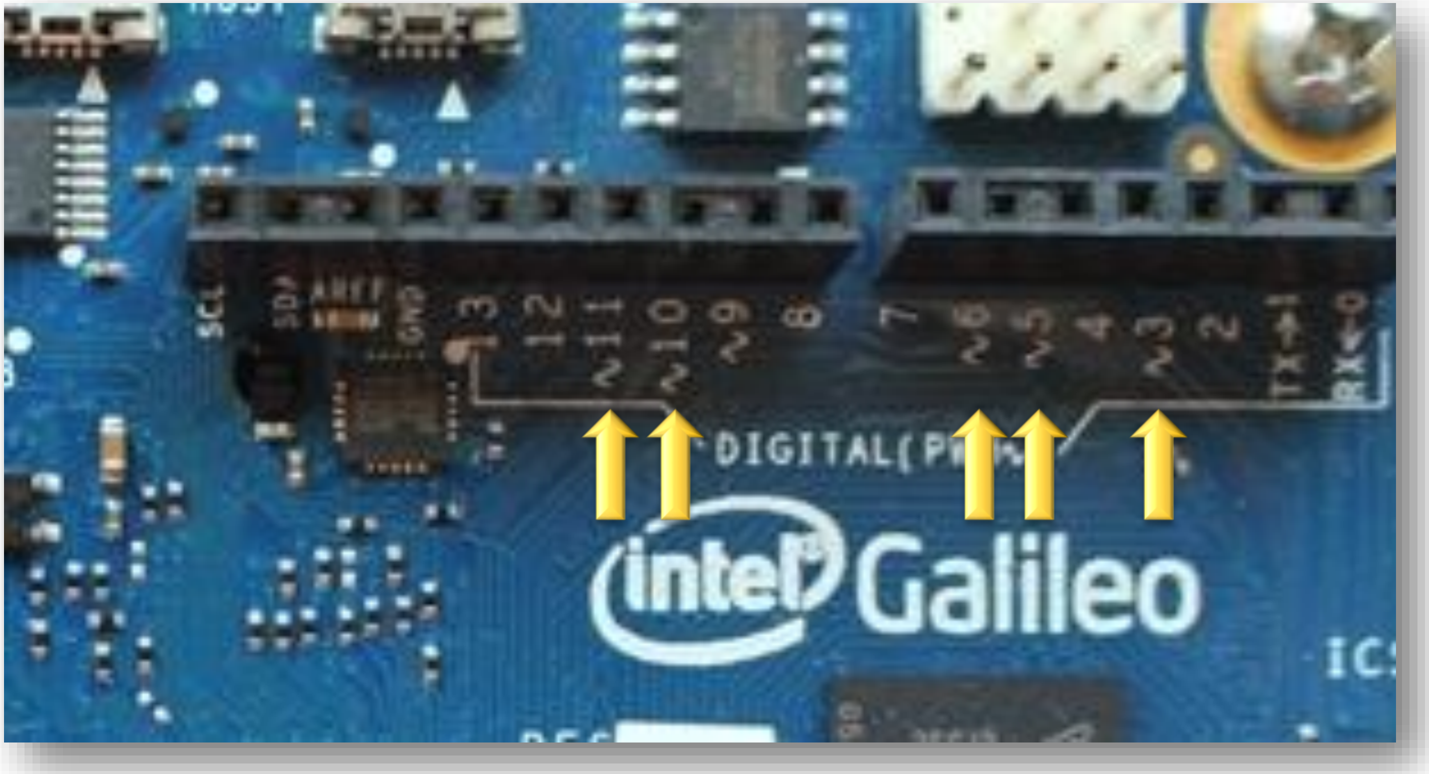
Analog Write



Pulse Width Modulation Out

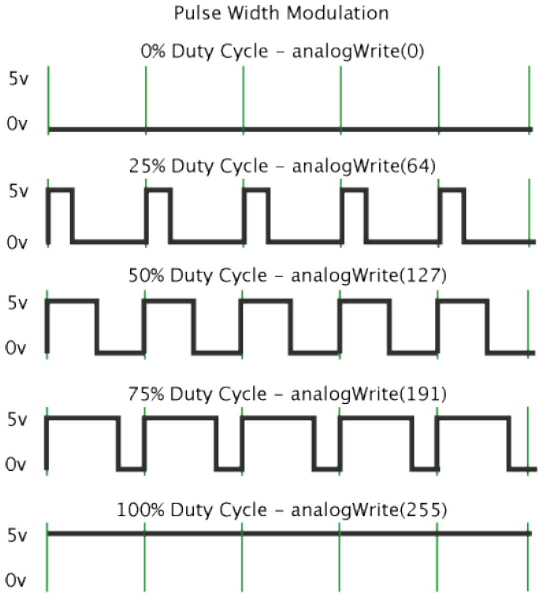
Pulse Width Modulation

- PWM is a technique for getting analog results with digital means
- Can be used to light a LED at varying brightness or drive a motor at various speeds
- PWM is supported on Pin 3, 5, 6, 9, 10, and 11



Pulse Width Modulation

- Digital control is used to create a square wave, a signal switched between on and off.
- This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends “On” (5volts) versus “Off” (0volts)
- To get varying analog values, you change, or modulate, that pulse width
- A call to analogWrite() is on a scale of 0 – 255…
  - analogWrite of 255 requests a 100% duty cycle (always on)
  - analogWrite of 127 is a 50% duty cycle (on half the time)
  - analogWrite of 0 is a 0% duty cycle (Always off)


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

Source: http://arduino.cc/en/Tutorial/PWM

Analog Write – Key Concepts

Reminder pinMode(): Configures the specified pin to behave either as an input or an output

- Syntax: pinMode(pin, mode)
- Example: pinMode(11, OUTPUT) set pin 11 to output mode

analogWrite(): Writes an analog value (Pulse Width Modulation wave) to a pin.

- Syntax: analogWrite(pin, value)
- Example: analogWrite (11, 32) send PWM of 100 to pin 11
- What's the Duty Cycle if analogWrite is 32?

Serial Read – LED Fade

LAB

Project Name: LED Fade *(Instructor Lead)*

Objective: Change the brightness of an LED to make it appear to fade in and out

Software Elements

- analogWrite()
- pinMode()

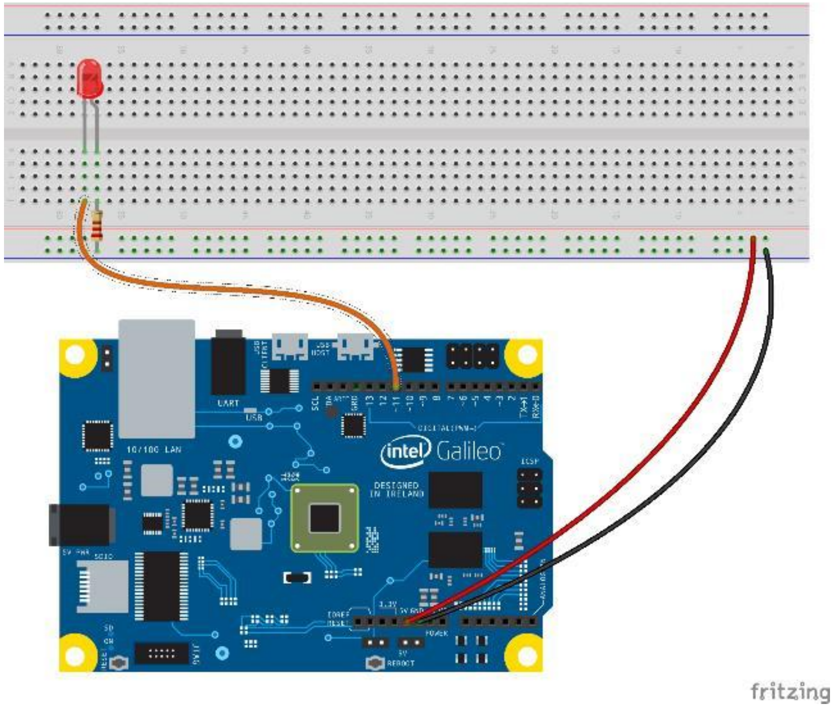
Components

- Breadboard (QTY1)
- Jumper Wires
- 5mm LED (Qty 1)
- 220 Ohm Resister (Qty1)

Analog Write – Fade LED

LAB

```
/*
 Fade

 This example shows how to fade an LED on pin 11
 using the analogWrite() function.

 This example code is in the public domain.
 */

int led = 11;           // the pin that the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by

// the setup routine runs once when you press reset:
void setup()  {
  // declare pin 11 to be an output:
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()  {
  // set the brightness of pin 9:
  analogWrite(led, brightness);

  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;

  // reverse the direction of the fading at the ends of the fade:
  if (brightness == 0 || brightness == 255) {
    fadeAmount = -fadeAmount ;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```

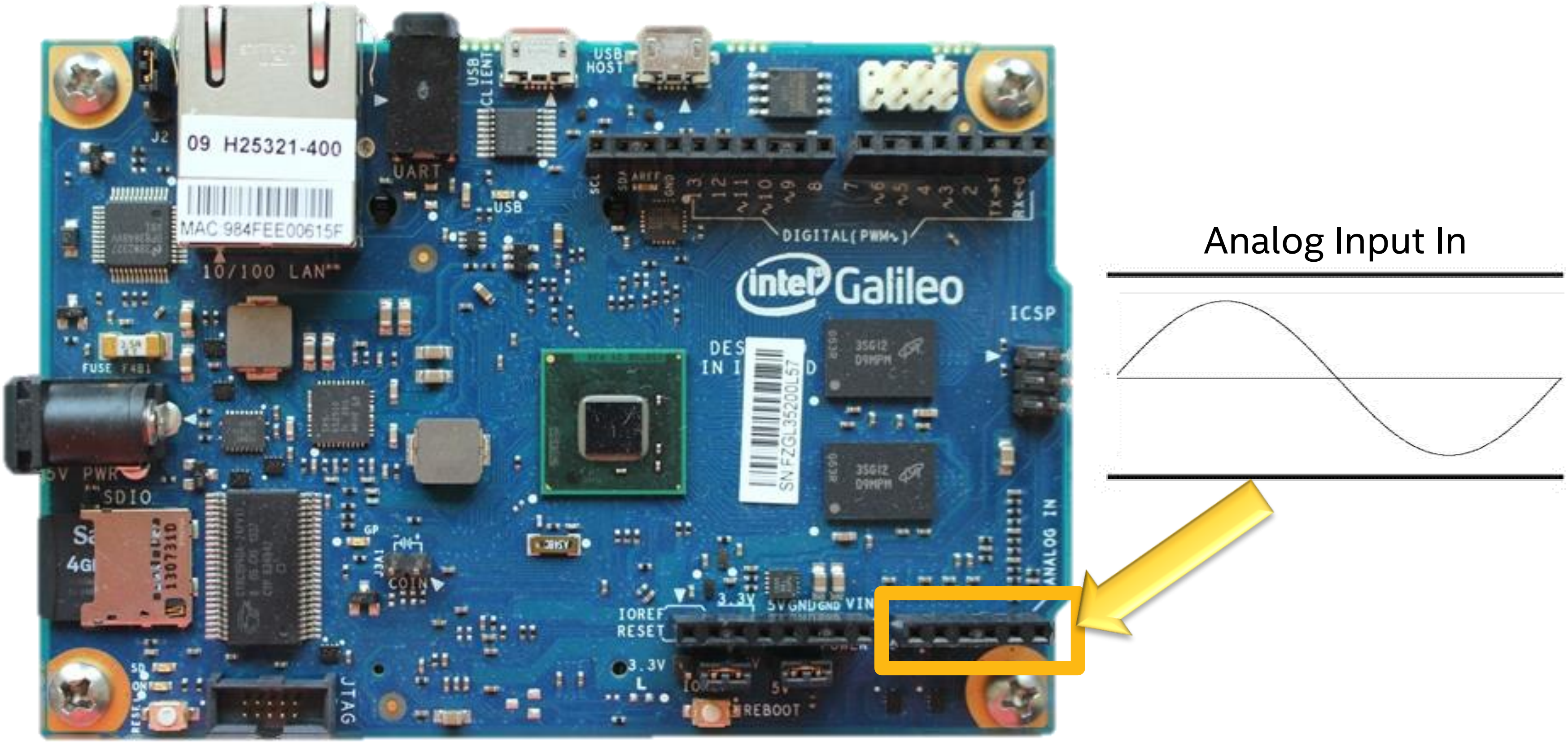


* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section5-AnalogWrite\Blink_Fade

Basic I/O Analog Read

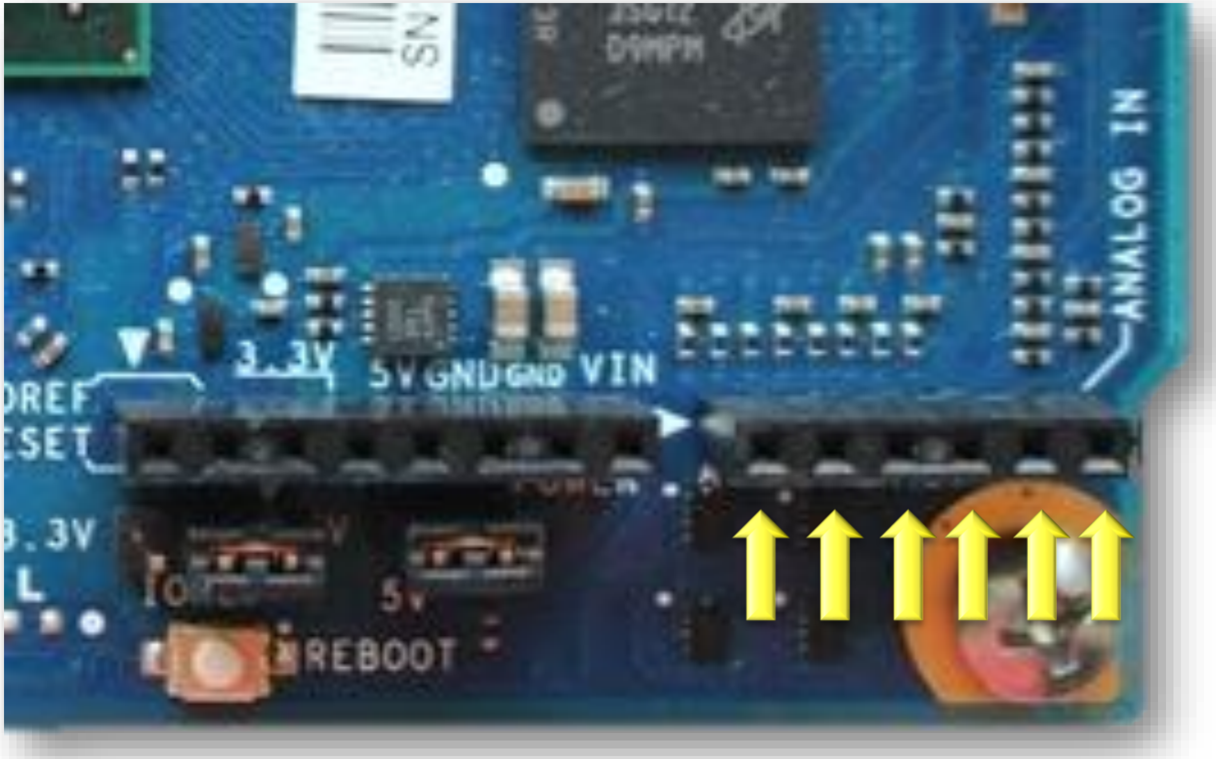
Analog Read



Analog Input In

Analog Pins

- Analog Pins are A0-A5
- Will map input voltages between 0 and 5 volts into integer values between 0 and 1023
- Can also be used as digital pins: analog input 0 as digital pin 14 through analog input 5 as digital pin 19



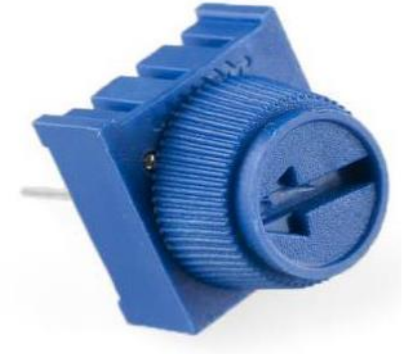
Analog Read – Key Concepts

analogRead(): Reads the value from the specified analog pin.

- Syntax: analogRead(pin)
- Example: analogRead(A0) reads analog value of pin A0

map(): Re-maps a number from one range to another. Value can be mapped to values that are out of range.

- Syntax: map(Source, fromLow, fromHigh, toLow, toHigh)
- Example: map(val, 0, 1023, 0, 254)



Analog Read – POT Value Read

LAB

Project Name: Reading Analog Values *(Instructor Lead)*

Objective: Read POT value and map to alternative range

Software Elements

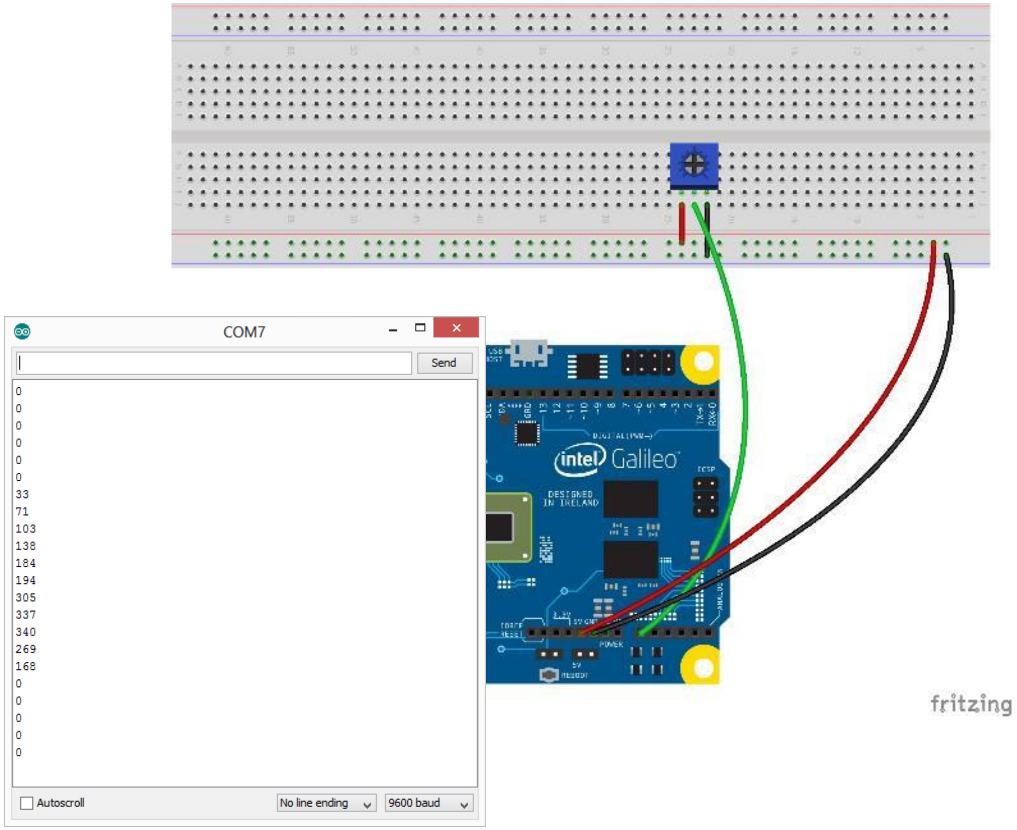
- analogRead()
- Serial.println()
- map()

Components

- Breadboard (QTY1)
- Jumper Wires
- Potentiometer

Analog Read – Read and Write POT Value

```
/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 This example code is in the public domain.
 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
}

// the loop routine runs over and over again forever:
void loop() {
  // read the input on analog pin 0:
  int sensorValue = analogRead(A0);
  // print out the value you read:
  Serial.println(sensorValue);
  delay(500);        // delay in between reads for stability
}
```



COM7

```
0
0
0
0
0
0
33
71
103
138
184
194
305
337
340
269
168
0
0
0
0
0
```

□ Autoscroll

No line ending

9600 baud

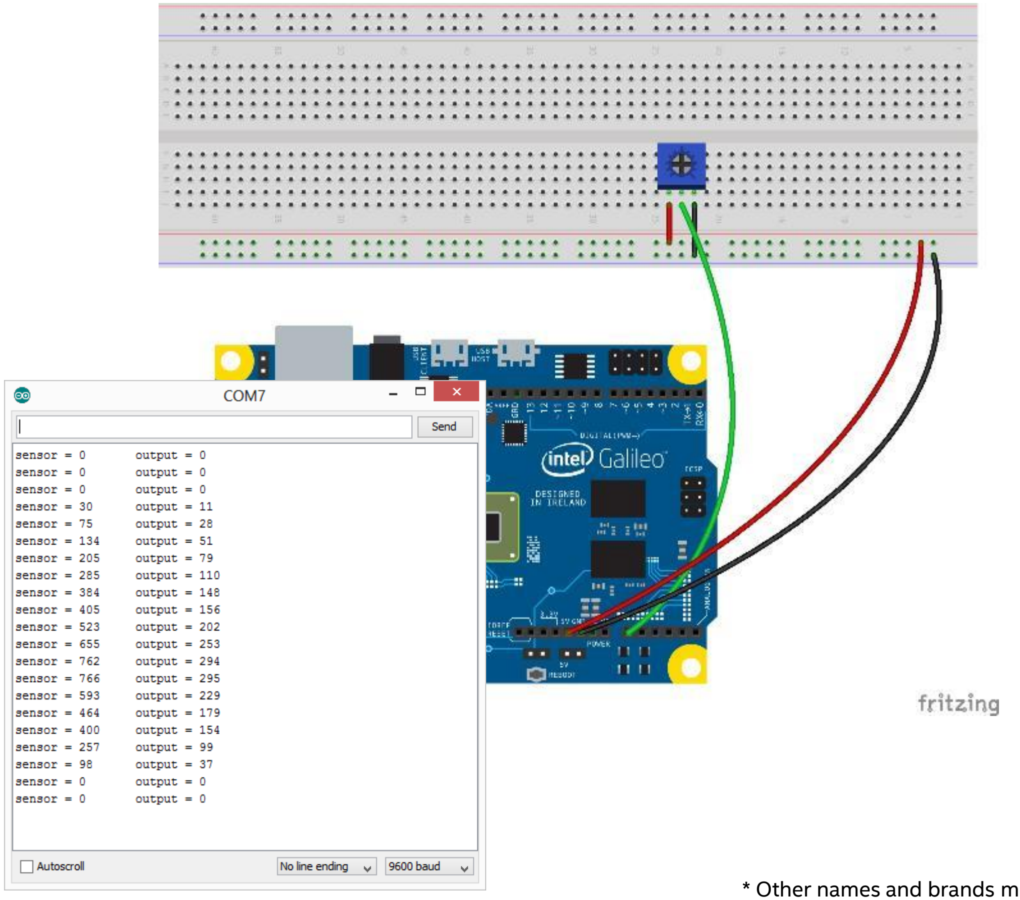
* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section6-AnalogRead\AnalogReadSerial

Analog Read – Read and Write Map Value

LAB

```
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
//const int analogOutPin = 11; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
//  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop
  // for the analog-to-digital converter to settle
  // after the last reading:
  delay(200);
}
```



```
sensor = 0      output = 0
sensor = 0      output = 0
sensor = 0      output = 0
sensor = 30     output = 11
sensor = 75     output = 28
sensor = 134    output = 51
sensor = 205    output = 79
sensor = 285    output = 110
sensor = 384    output = 148
sensor = 405    output = 156
sensor = 523    output = 202
sensor = 655    output = 253
sensor = 762    output = 294
sensor = 766    output = 295
sensor = 593    output = 229
sensor = 464    output = 179
sensor = 400    output = 154
sensor = 257    output = 99
sensor = 98     output = 37
sensor = 0      output = 0
sensor = 0      output = 0
```

□ Autoscroll   No line ending   9600 baud

* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section6-AnalogRead\AnalogReadSerialMap

Analog Write- Engineering Challenge

LAB

Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

Challenge 1: Fade LED based on POT Value

Engineering Challenge Review

Challenge Review

Challenge 1: Fade LED based on POT Value

```
// These constants won't change.  They're used to give names
// to the pins used:
const int analogInPin = A0;  // Analog input pin that the potentiometer is attached to
const int analogOutPin = 11; // Analog output pin that the LED is attached to

int sensorValue = 0;        // value read from the pot
int outputValue = 0;        // value output to the PWM (analog out)

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  // read the analog in value:
  sensorValue = analogRead(analogInPin);
  // map it to the range of the analog out:
  outputValue = map(sensorValue, 0, 1023, 0, 255);
  // change the analog out value:
  analogWrite(analogOutPin, outputValue);

  // print the results to the serial monitor:
  Serial.print("sensor = " );
  Serial.print(sensorValue);
  Serial.print("\t output = ");
  Serial.println(outputValue);

  // wait 2 milliseconds before the next loop for the analog-to-digital converter to settle after the last reading:
  delay(2);
}
```

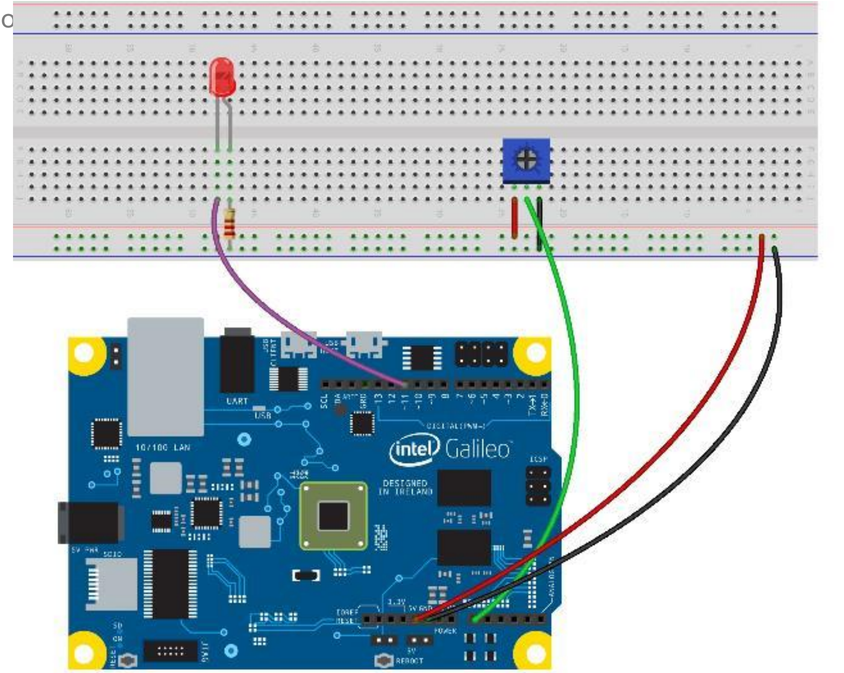


Image Source from Fritzing*

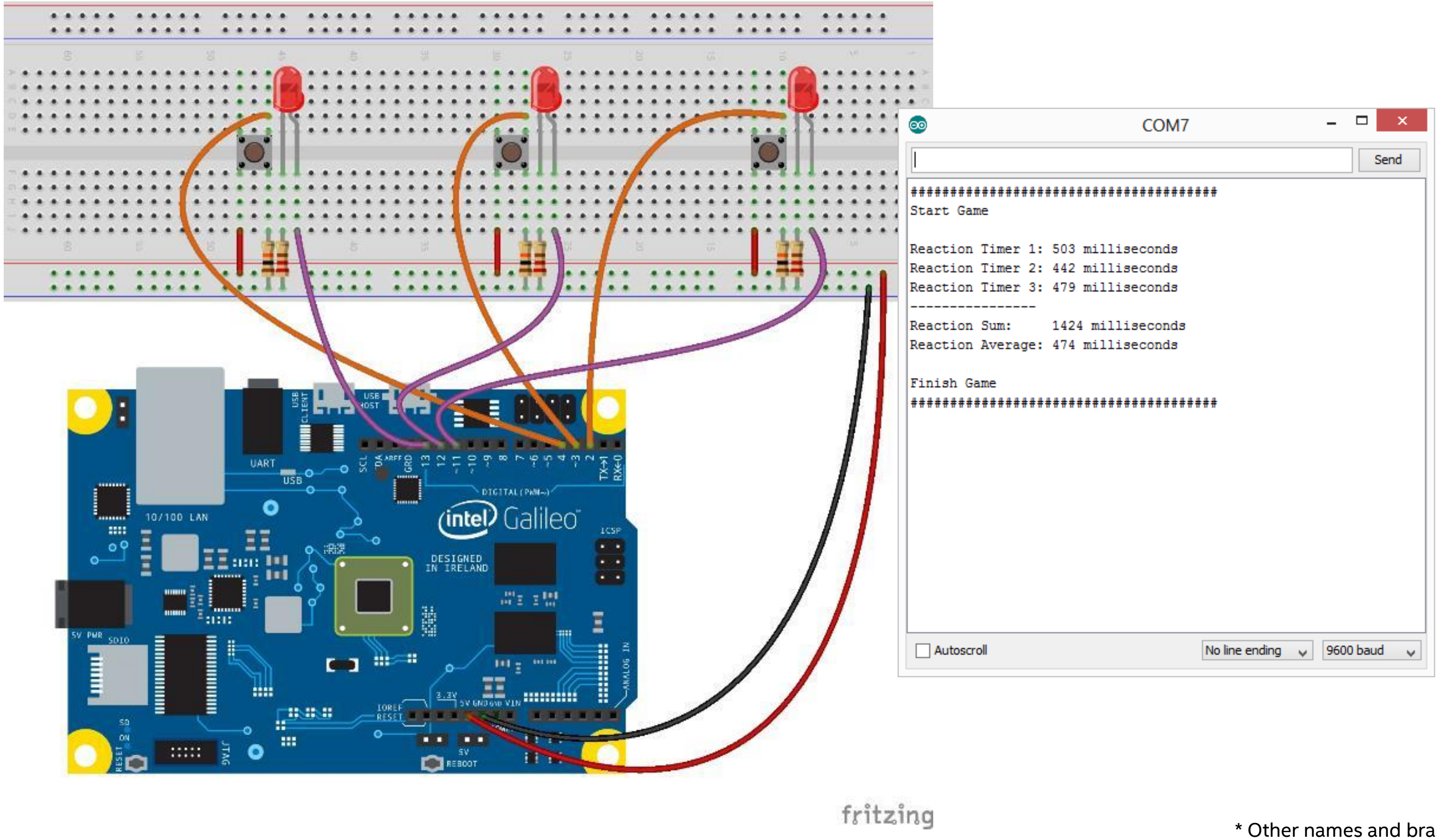
Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section6-AnalogRead\AnalogReadMapFade

* Other names and brands may be claimed as the property of others

Section Review Challenge

Reaction Timer

Source Code: *Lesson2-BasicIO\Section7-Review\ReactionTimerInital*



```
########################################
Start Game

Reaction Timer 1: 503 milliseconds
Reaction Timer 2: 442 milliseconds
Reaction Timer 3: 479 milliseconds
----------------
Reaction Sum:     1424 milliseconds
Reaction Average: 474 milliseconds

Finish Game
########################################
```

* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section7-Review\ReactionTimerInital

Capstone - Engineering Challenge

LAB

Using your own circuit design and Arduino sketch, design a solution that solves the following challenges.

Use previous Lab example as needed for reference

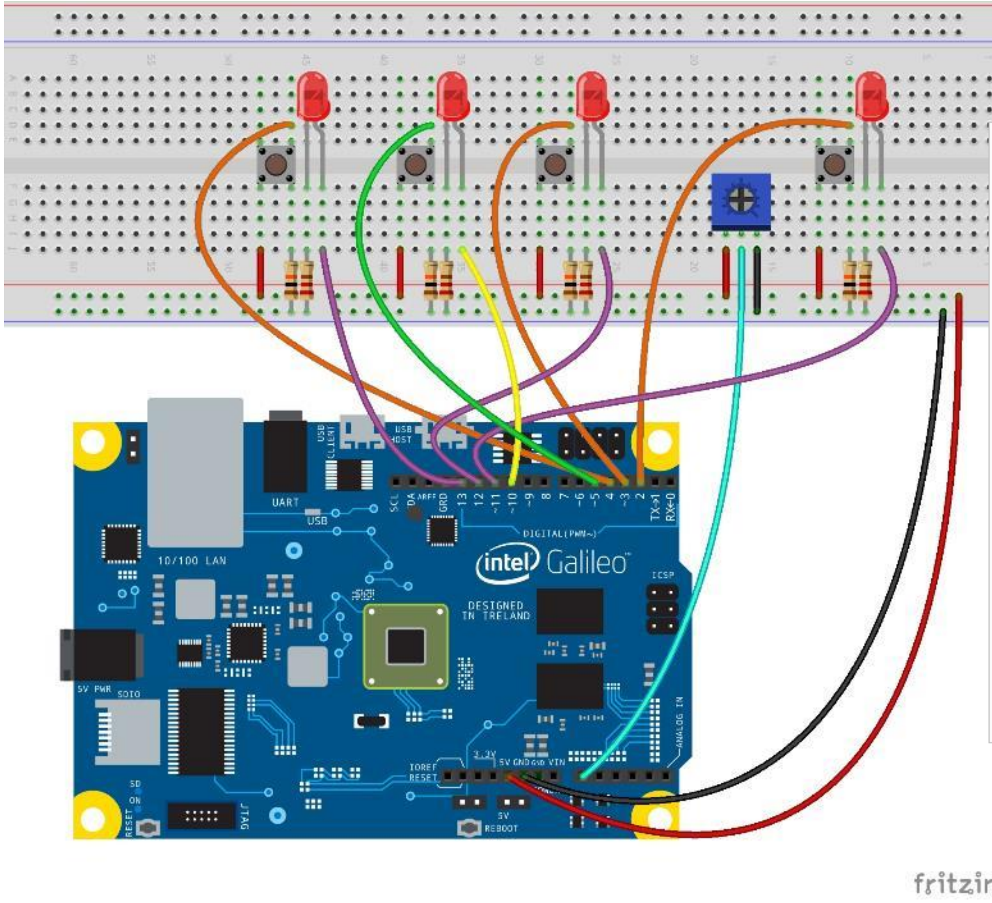
Challenge 1: Add additional Reaction Button and Indicator

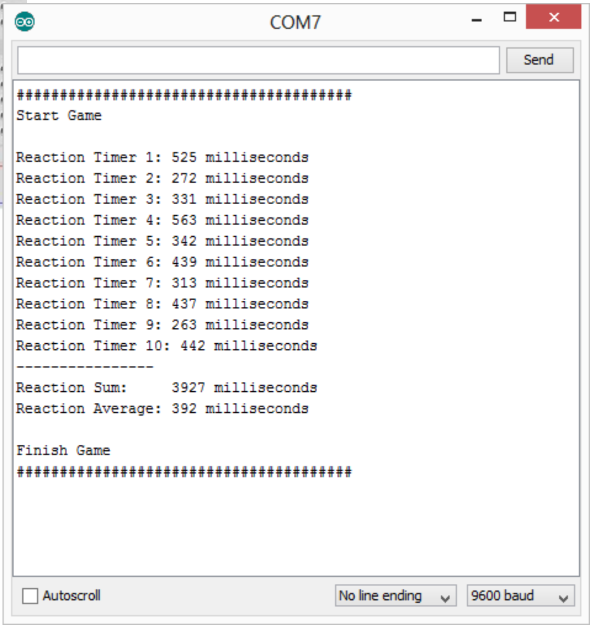
Challenge 2: Add variable knob to change the number of times the game ready indicator blinks before the start of game

Capstone – Reaction Timer

Challenge Review

Source Code: *Lesson2-BasicIO\Section7-Review\ReactionTimerChallenge*





```
#########################################
Start Game

Reaction Timer 1: 525 milliseconds
Reaction Timer 2: 272 milliseconds
Reaction Timer 3: 331 milliseconds
Reaction Timer 4: 563 milliseconds
Reaction Timer 5: 342 milliseconds
Reaction Timer 6: 439 milliseconds
Reaction Timer 7: 313 milliseconds
Reaction Timer 8: 437 milliseconds
Reaction Timer 9: 263 milliseconds
Reaction Timer 10: 442 milliseconds
----------------
Reaction Sum:     3927 milliseconds
Reaction Average: 392 milliseconds

Finish Game
#########################################
```

☐ Autoscroll   No line ending   9600 baud

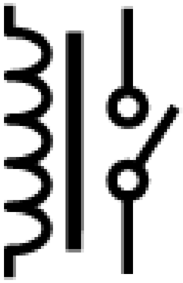
* Other names and brands may be claimed as the property of others

Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section7-Review\ReactionTimerChallenge

But wait there's more…

Reapplying Concepts Learned

digitalWrite +



Project Files - USB Drive:\Lessons\Lesson2-BasicIO\Section7-Capstone\RelayExample

Questions

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm%20 Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, visit Intel Performance Benchmark Limitations

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Celeron, Intel, Intel logo, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel SpeedStep, Intel XScale, Itanium, Pentium, Pentium Inside, VTune, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Intel® Active Management Technology requires the platform to have an Intel® AMT-enabled chipset, network hardware and software, as well as connection with a power source and a corporate network connection. With regard to notebooks, Intel AMT may not be available or certain capabilities may be limited over a host OS-based VPN or when connecting wirelessly, on battery power, sleeping, hibernating or powered off. For more information, see http://www.intel.com/technology/iamt.

64-bit computing on Intel architecture requires a computer system with a processor, chipset, BIOS, operating system, device drivers and applications enabled for Intel® 64 architecture. Performance will vary depending on your hardware and software configurations. Consult with your system vendor for more information.

No computer system can provide absolute security under all conditions. Intel® Trusted Execution Technology is a security technology under development by Intel and requires for operation a computer system with Intel® Virtualization Technology, an Intel Trusted Execution Technology-enabled processor, chipset, BIOS, Authenticated Code Modules, and an Intel or other compatible measured virtual machine monitor. In addition, Intel Trusted Execution Technology requires the system to contain a TPMv1.2 as defined by the Trusted Computing Group and specific software for some uses. See http://www.intel.com/technology/security/ for more information.

†Hyper-Threading Technology (HT Technology) requires a computer system with an Intel® Pentium® 4 Processor supporting HT Technology and an HT Technology-enabled chipset, BIOS, and operating system. Performance will vary depending on the specific hardware and software you use. See www.intel.com/products/ht/hyperthreading_more.htm for more information including details on which processors support HT Technology.

Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, virtual machine monitor (VMM) and, for some uses, certain platform software enabled for it. Functionality, performance or other benefits will vary depending on hardware and software configurations and may require a BIOS update. Software applications may not be compatible with all operating systems. Please check with your application vendor.

* Other names and brands may be claimed as the property of others.

Other vendors are listed by Intel as a convenience to Intel's general customer base, but Intel does not make any representations or warranties whatsoever regarding quality, reliability, functionality, or compatibility of these devices. This list and/or these devices may be subject to change without notice.

Copyright © 2014, Intel Corporation. All rights reserved.

intel®

Backup