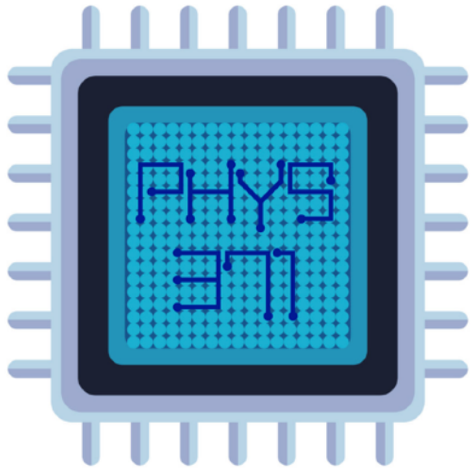


COURSE INTRODUCTION

Dr. Riccardo Longo

01/20/2023

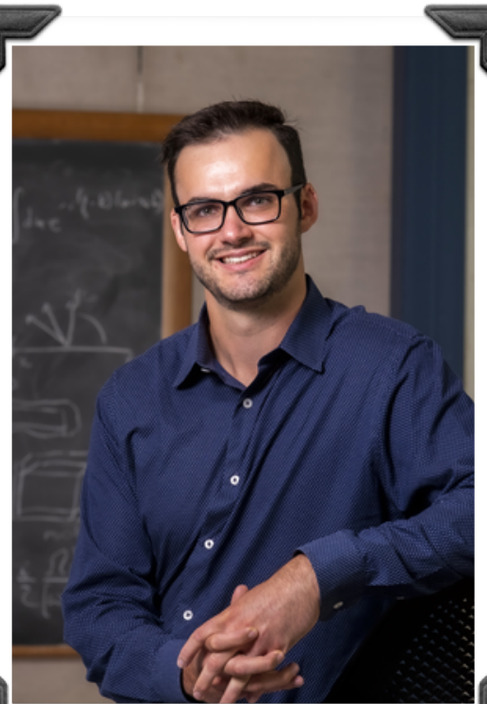
Lecture 1

PHYS371 - People (and credits)

Instructor for Spring 2023:
Dr. Riccardo Longo
rlongo@illinois.edu

Former Instructor & main author of material available for the course (thanks!)
Prof. George Gollin
g-gollin@illinois.edu

TA for Spring 2023:
Matthew Hoppesch
mch6@illinois.edu



TA for Spring 2023:
Jenny Campbell
jjc11@illinois.edu

PHYS371 - Contacts

Instructor for Spring 2023:
Dr. Riccardo Longo
rlongo@illinois.edu

- Please use **[PHYS371] as a prefix** in the subject of your emails to us related to the course.
- Please keep all the three of us in the recipients.
- In case of e-mail threads, do not forget to “reply-all”

TA for Spring 2023:
Matthew Hoppesch
mch6@illinois.edu

TA for Spring 2023:
Jenny Campbell
jjc11@illinois.edu

PHYS371 - Course Goals

- During this course, you will get in touch with a lot of new concepts, hardware, hands-on work
- While navigating through them, the goal for you is to achieve a few main goals
  - Learn how to tackle a problem and set up a measurement you could make
  - Learn how to find resources and use them to carry out an experiment
  - Learn how to construct and build a device that might allow you to make the measurement
  - Learn how to commission and calibrate your device
  - Learn how to carry out data-taking and analysis
- In other words, you will develop a number of on-field soft skills that will help you in the continuation of your career

PHYS371 - Course Goals

- During this course, you will get in touch with a lot of new concepts, hardware, hands-on work
- While navigating through them, the goal for you is to achieve a few main goals
  - Learn how to tackle a problem and set up a measurement you could make
  - Learn how to find resources and use them to carry out an experiment
  - Learn how to construct and build a device that might allow you to make the measurement
  - Learn how to commission and calibrate your device
  - Learn how to carry out data-taking and analysis
- In other words, you will develop a number of on-field soft skills that will help you in the continuation of your career

But, most importantly, you will learn how to ask yourself critical questions

Is this instrument adequate to carry out this measurement?

Does this result make sense?

Am I looking at a physics effect or a systematic of the measurement?

What is this result telling me?

Is this a problem someone else may have already tackled?

PHYS371 - Course (Software) Tools

You will use different software - here are the main ones that you must install (or register on)

Arduino IDE

Arduino IDE
https://www.arduino.cc/en/software

TINKERCAD AUTODESK® TINKERCAD®

https://www.tinkercad.com/
Click on: this link and login with your NetID
All the students should be able to login.
If not the case, please send me an email !

ANACONDA® SPYDER The Scientific Python Development Environment

Anaconda Spider IDE
available through Anaconda

GitLab

https://gitlab.engr.illinois.edu/
Each group will have to create a code repository on GitLab, with all the members, the instructor (rlongo) and the TAs (mch6, jjc11) added as developers

PHYS371 - Course Structure

- Mostly hands-on class, w/ some mini-seminars (up to 20 min) in the first 3-4 lectures of the course
  - **Week 1 [today]:** Intro to the course, group formation, intro and first operations w/ Arduino
  - Week 2: From python to C++: how to code w/ Arduino
  - Week 4: Introduction to Fusion360 (more advanced than ThinkerCad) from D.MacLean [UIUC and PHYS371 alumnus - now working as a junior engineer at FNAL]
  - Week 5: Project presentation and workplan from each group [preparation of it will be the homework]
- Homework to get you started with Arduino, help you to get your project on track, and get a bit more in-depth w/ lecture content will be assigned in the first half of the course.
  - Assignments from one week to the other will be illustrated at the end of each lecture on Friday
  - 1 or 2 groups per week will also prepare a short presentation (~10 min) on a particular sensor (or microcontroller, or communication standard) to illustrate it to the rest of the class
  - For presentations, see **"How to present a Physics Talk"**, by Celia Elliot

PHYS371 - Course Requirements

- Homework assignments turned in on time
- Teamwork to move forward with the project with your group mates, and with good bookkeeping
  - Code stored in a **GitLab** repository named '*Group-XX-PHYS371-SP2023*' (group # assigned today)
  - Code commented properly (a bit more about coding in the next lecture)
  - Logging of activities in a logbook (paper or electronic)
  - Task sharing - but keeping a detailed understanding of all the work done by the group
- Reports during the Friday lecture to your course mates (preparation will be part of the homework)
  - Reports can be prepared in PowerPoint or Keynote, depending on your preferred tool within the group.
  - Reports have to be accessible to the audience
- In place of a final exam, you will have to turn in two documents:
  - A **30' minute PowerPoint (or Keynote) presentation** describing your project. Your intended audience comprises your Physics 371 classmates, who will question you at the end of the presentation. Presentations are due at the start of class, week 14; you will present during the 14th week of the term.
  - A **written report** of 15-20 pages, single-spaced, describing your measurements and conclusions; this is essentially the same information you will write into your presentation. However, your report should target an audience outside the university community. Your report is due in class on week 14.

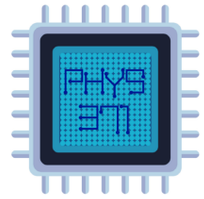
PHYS371 - Grading

The final grading will be individual, e.g. individuals from the same group can receive different grades.

The grade will be computed taking into account

- Quality of your submitted homework assignments
- Attendance in classroom and meetings with Instructor&TAs
    - You are required to attend each and every one of the course meetings, arriving on time with your laptop computer, charger, notebook, and device hardware. Excused absences will be granted and documented in accordance with University policy as described in Article 1, Part 5 Class Attendance, of the Student Code.
    - You **must** file your documentation concerning an excused absence on the Physics Department's Excused absences portal within **two weeks** of your absence.
        - The final grade will be reduced by half a letter grade per unexcused absence.
        - Please consult **the course webpage** for more details and ask the Instructor if any clarification is needed
- Quality and efficiency of your group's collaborative interaction
- Quality of your project hardware and data acquisition system
- Quality of your offline analysis software work, and proper code bookkeeping on GitLab;
- Participation in scientific discussions held in class (e.g. Q&A sessions
- Independency & creativity demonstrated during the development of the project
- Compliance with meeting times and with deadline
- Quality of presentations during the course, final presentation and final report

PHYS371 - Other notes



Homework Assignments Wildcard (1 per student)

Can be used to 1. turn-in one assignment with a delay up to one week (e.g. by the due date of the next week assignment) or 2. re-try an assignment where the grade received is unsatisfactory. In this case, the homework will be returned and the student will have until the next homework due date to re-work it. The wildcard can be used only one time during the course for either case 1. or 2., by e-mailing the TAs and me by the due date of the assignment.

Access to irrelevant content during lectures

For our class you are not to access anything that is not directly relevant to the work at hand: no visits to social media sites, or unnecessary logins to your messaging apps.

Academic integrity

You must never submit the work of someone else as your own. Always identify your sources.

It is cheating to receive work from another student and then represent it as your own. It is cheating—and a violation of U.S. copyright law—to give (or sell) course material to someone else who intends to redistribute and/or sell it.

Cheating will be penalized harshly. See course website for more details.

Dr. Riccardo Longo

01/20/2023

Lecture 1

Possible 30’ time slots for meetings

Availabilities of the full course staff (Instructor + TAs)

Monday: 10 AM to noon, 13:30 PM to 14:30 PM

Tuesday: 4 PM to 6 PM

Wednesday: 11 AM to noon, 3:30 PM to 6 PM

Thursday: 11 AM to noon

Availabilities of the Instructor + one TA

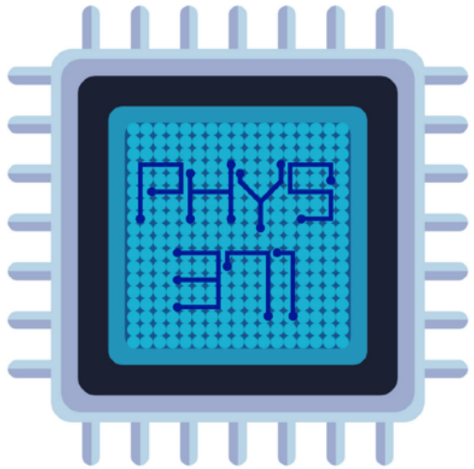
Tuesday: 13:30 PM to 4 PM

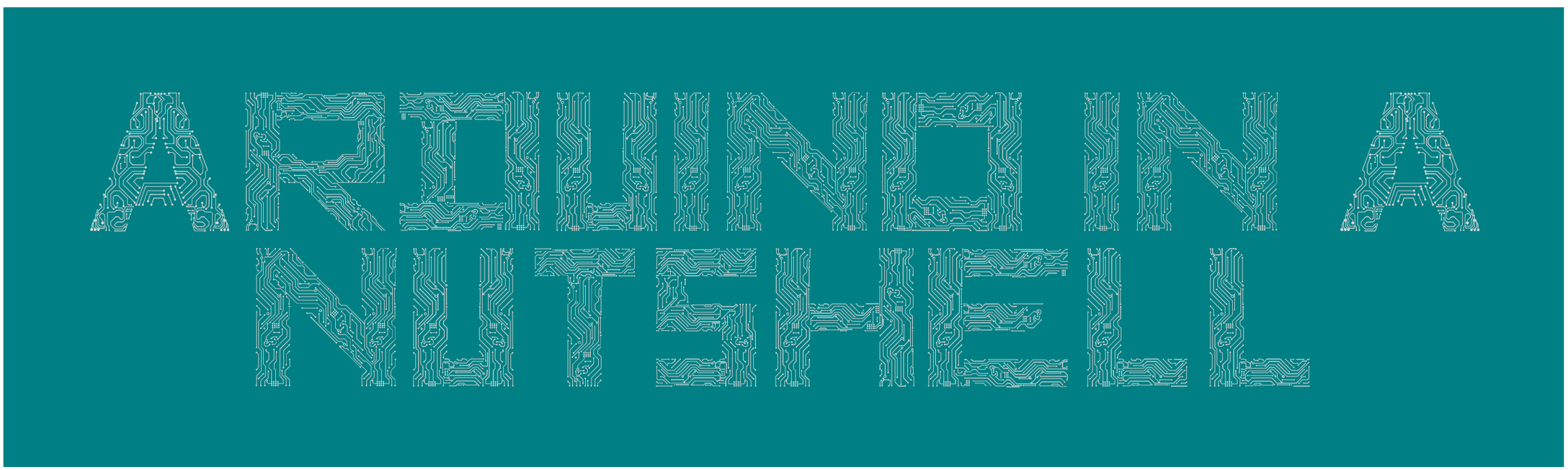
Thursday: 1 PM to 4 PM

Location of each meeting will be decided w/ each group once the time slot is fixed

Time and location will be then indicated on the course webpage

START TIME	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
8:00 AM							
8:30 AM							
9:00 AM							
9:30 AM							
10:00 AM							
10:30 AM							
11:00 AM							
11:30 AM							
12:00 PM							
12:30 PM							
1:00 PM							
1:30 PM							
2:00 PM							
2:30 PM							
3:00 PM							
3:30 PM							
4:00 PM							
4:30 PM							
5:00 PM							
5:30 PM							
6:00 PM							
6:30 PM							





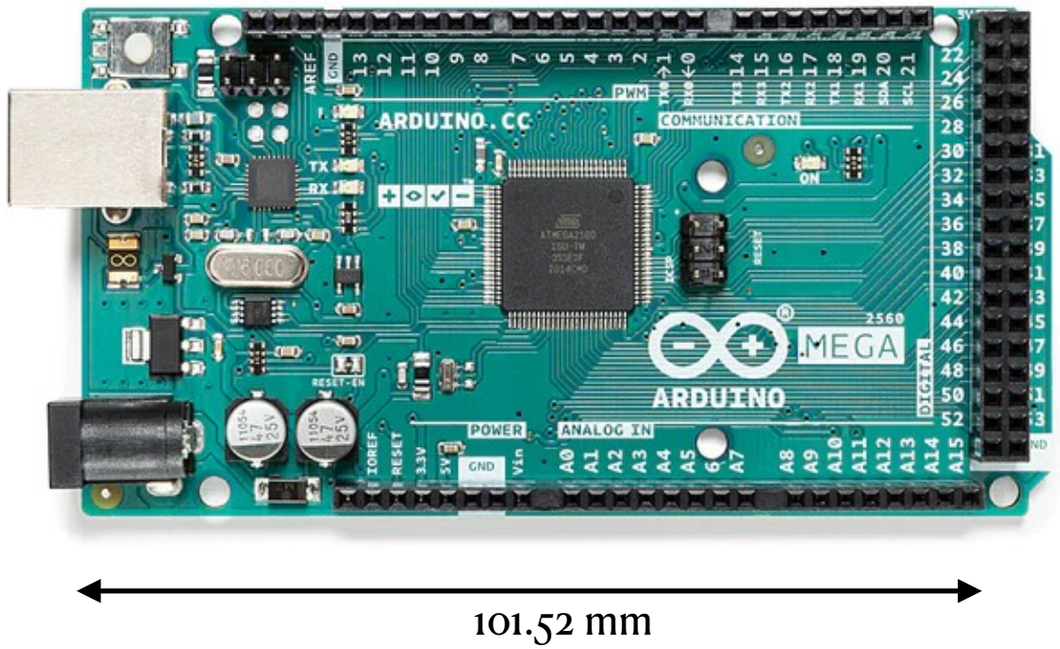
An (incomplete but) introductory overview

Dr. Riccardo Longo

01/20/2023

Lecture 1

Arduino Mega 2560

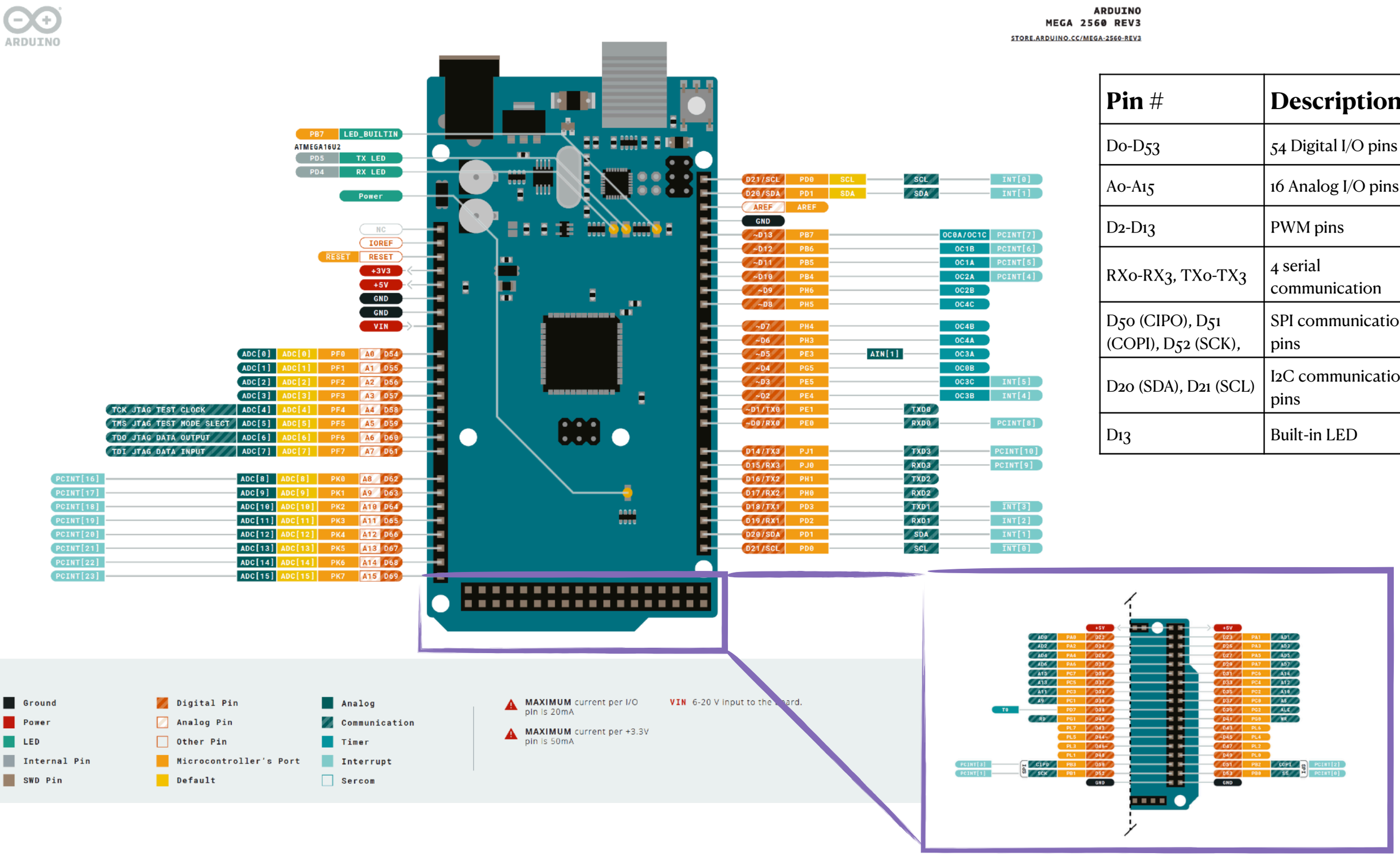


Operating Voltage	5V
Input Voltage (Rec.)	7-12V
Input Voltage (Lim.)	6-20V
DC current per I/O pin	20 mA
DC current for 3.3V pin	50 mA
Flash memory	256 kB
SRAM	8 kB
Clock Speed	16 MHz
Weight	37 g

- Based on **ATMega2560** microprocessor
- 54 digital input/output pins (15 can be used as Pulse width modulation (PWM) output)
- 16 analog inputs
- 4 Universal Asynchronous Receiver-Transmitter ports (UARTs, hardware serial ports)
- One 16 MHz crystal oscillator (🕰️)
- One USB connection
- One power jack
- One In-Circuit Serial Programming header
- One USB-B port

All this for... **48.37 $** (!!!)

Arduino Mega 2560 - a more detailed view...



Pin #	Description
D0-D53	54 Digital I/O pins
A0-A15	16 Analog I/O pins
D2-D13	PWM pins
RX0-RX3, TX0-TX3	4 serial communication
D50 (CIPO), D51 (COPI), D52 (SCK),	SPI communication pins
D20 (SDA), D21 (SCL)	I2C communication pins
D13	Built-in LED

Signal transmission

- In a digital signal, data is transferred as a sequence of *high to low* and *low to high* switchings that occur very rapidly.
- High: 1, Low: 0 —> **bits**! (Binary system)
- Sequence of 8 bits —> **byte**!

Example

1	0	1	1	1	0	0	1

$$\begin{array}{cccccccc} \times & \times & \times & \times & \times & \times & \times & \times \\ 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\ = & = & = & = & = & = & = & = \\ 128 & +0 & +32 & +16 & +8 & +0 & +0 & +1 = 185 \end{array}$$

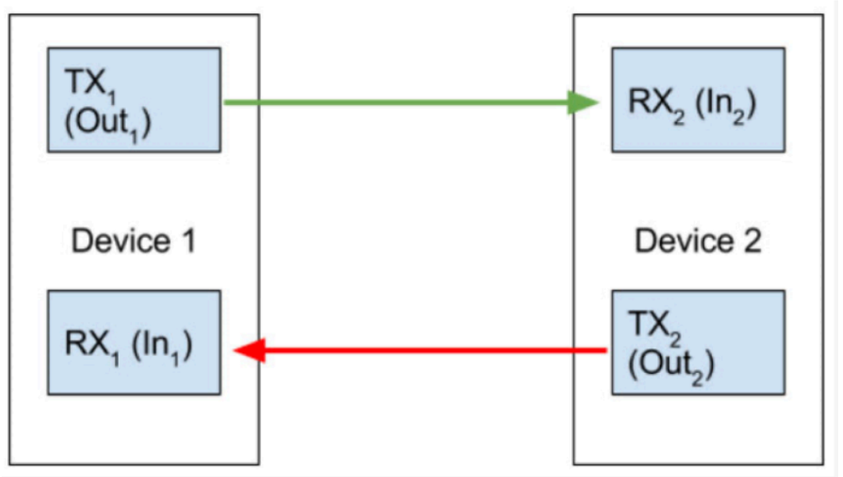
Convention

In software, binary numbers are prefixed with *0b*, octal numbers are prefixed with a *0*, and hexadecimal numbers are prefixed with a *0x*. Decimal numbers are not prefixed.

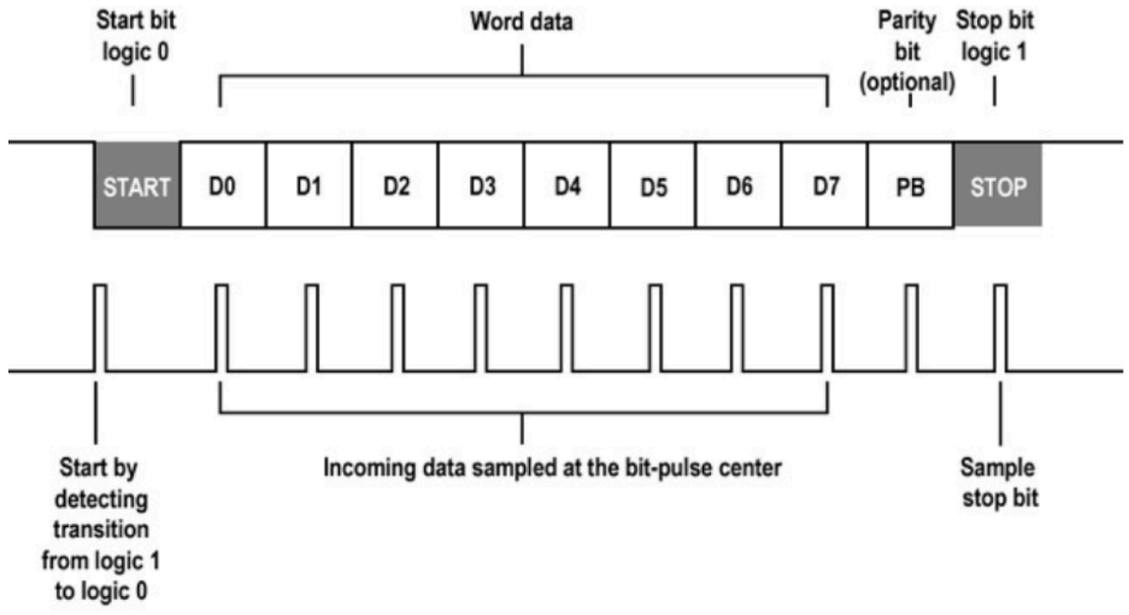
- The electrical engineering community decided to standardize electronics around three main communication protocols to ensure device compatibility.
  - Universal Asynchronous Receiver/Transmitter (UART)
  - Serial to Peripheral Interface (SPI)
  - Inter Integrated Circuit (I^2C)

UART communication principle

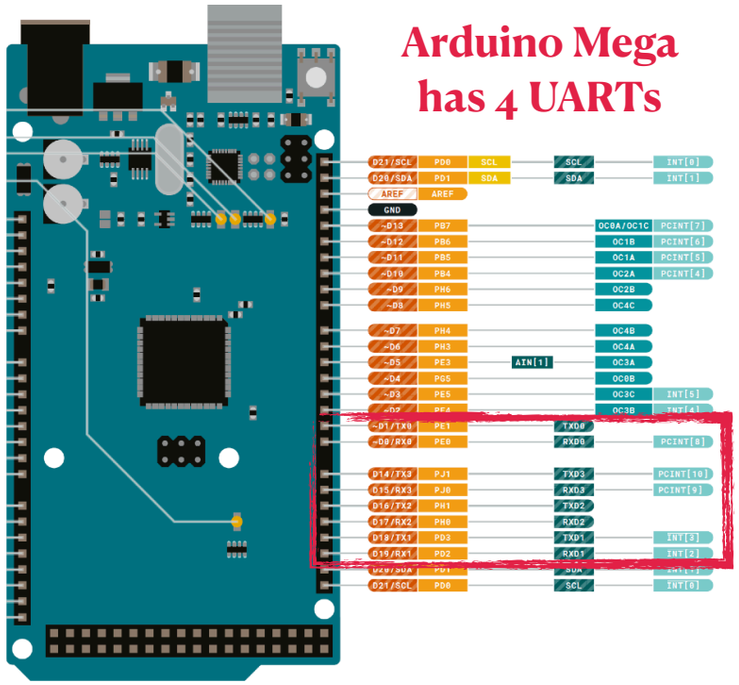
- Serial communication type - data transmitted as sequential bits
- One line for data transmission (TX) and one for data reception (RX)
- UART is the name of the onboard hardware that manages the packaging and translation of serial data.
- Why *Asynchronous*? because the communication does not depend on a synchronized clock signal between the two devices attempting to communicate with each other.



- Devices that communicate via UART send packets of pre-defined size that contain flags for the start and end of the message and confirmation of whether the message was received correctly.
- For this reason, UART is slower compared to synchronized forms of communications, since part of the stream is communication itself!
- UART is designed for communication between only two devices at a time.
- UART is *half-duplex*, which means that even though communication can occur bidirectionally, both devices cannot transmit data to each other at the same time



UART on Arduino

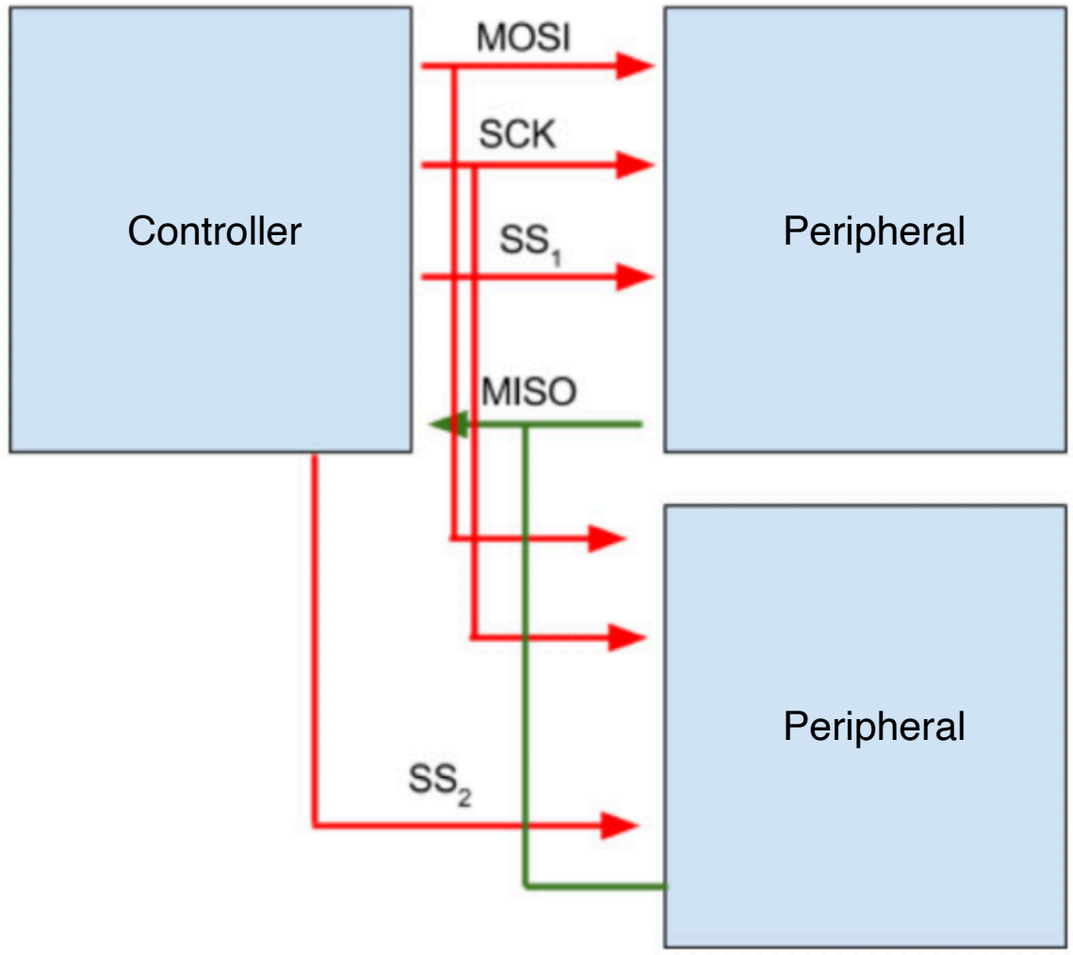


- How do we use UART w/ Arduino? Do we have to setup the communication at bit level?
  - No! Arduino provides higher level software libraries (e.g. *Serial* and *SoftwareSerial*) that are the only aspect of the communication process the user has to deal with.

Method	Purpose	Example Code	Explanation
Constructor (SoftwareSerial only)	Define the GPIO pins that will serve as the UART RX and TX lines	SoftwareSerial comms (2 , 3);	Defines a serial connection with RX line on GPIO 2 and TX line on GPIO 3.
begin	Define the baud rate (transmission speed) for the serial connection in range 4800 to 115200	comms.begin(9600);	Communication on the "comms" serial port will occur at 9600 baud.
print	Write byte data converted into human-readable characters over serial connection	comms.println("Hello World");	Writes bytes equivalent to *Hello World* (human-readable characters)
write	Write raw byte data over the serial connection	comms.write(45);	Writes byte with value 45.
available	Evaluates to true when data is available over the serial connection	if (comms.available())	Enter *if* statement if there is data available to read over the serial connection
read	Read data from the serial connection	comms.read();	Reads from serial connection

SPI communication principle

- Differences from UART:
  - Synchronous - e.g. tied to a clock signal
  - Follows a controller-responder model
  - *Full-duplex*, communication always occur bidirectionally
    - Makes use of of a shift-register
  - Requires more than 2 lines for implementation
    - ➡ COPI = "Controller Out Peripheral In" (common transmitter line)
    - ➡ SCK = "Clock" (common clock line)
    - ➡ CS = "Chip Select" (one per peripheral - selects communication)
    - ➡ CIPO = "Controller In Peripheral Out" (common receiver line)



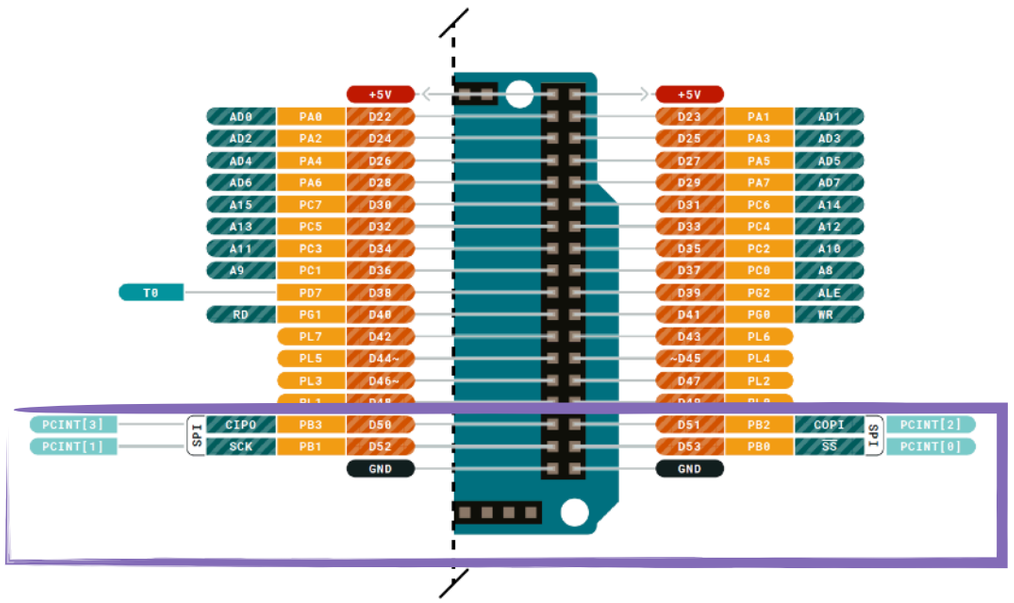
- Need of one general purpose input/output pin (GPIO) on the controller for each peripheral connected)

  CPOL = clock polarity - refers to the idle state of the clock signal (*low* or *high*)

  CPHA = clock phase - refers to the edge of the clock signal upon which data is captured (e.g. rising edge or falling edge)

	CPHA = 0 (rising edge)	**CPHA = 1 (Falling edge)**
CPOL = 0 (idle = low)	SPI mode 0	SPI mode 1
CPOL = 1 (Idle = high)	SPI mode 2	SPI mode 3

SPI on Arduino



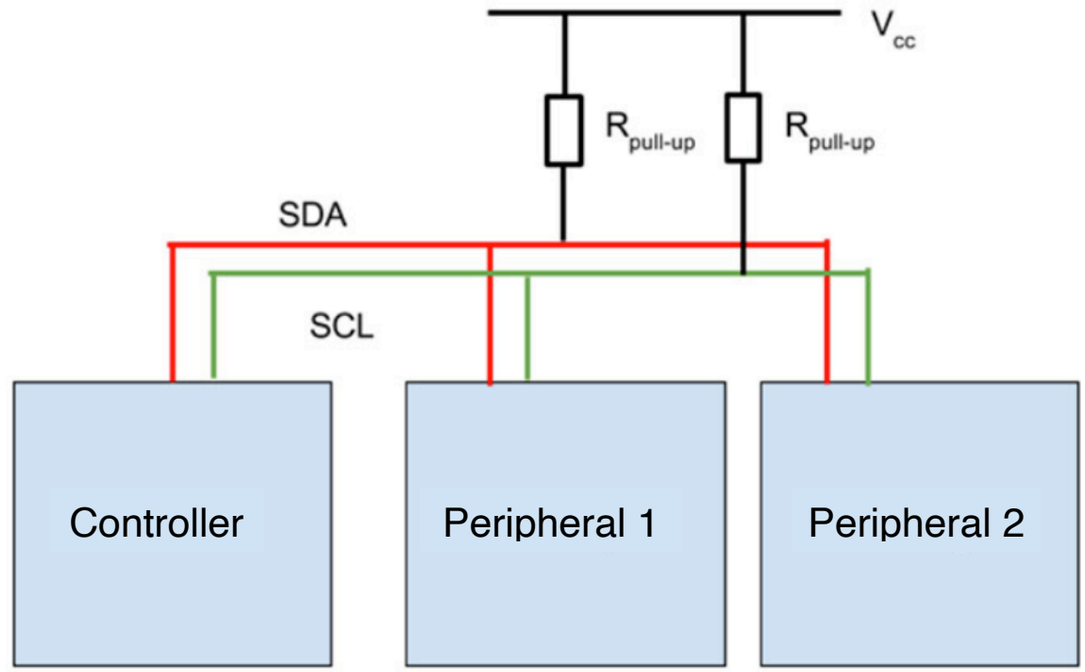
CIPO = D50
COPI = D51
SCK = D52
CS = D53 (or any other digital pin)

https://docs.arduino.cc/learn/communication/spi

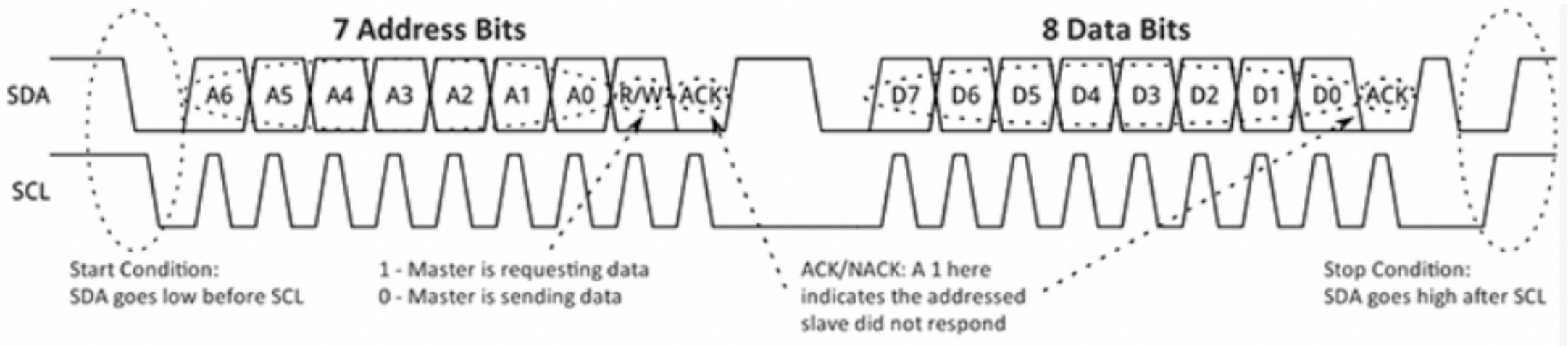
Method	Purpose	Example Code	Explanation
Constructor	Define the clock speed, data bit order (most significant bit first or least significant bit first), and SPI mode	`SPI.beginTransaction (SPISettings(14000000, MSBFIRST, SPI_MODE0));`	Defines a SPI connection at 14MHz in SPI Mode 0 and data transmission occurring with the most significant bit first
digitalWrite	Select the peripheral device attached to this GPIO pin	`digitalWrite(10, LOW);`	Drive the GPIO pin low to select the peripheral device. To de-select the device, the pin has to be driven high
transfer	Transfer the byte to the selected peripheral	`SPI.transfer(0x00);`	Send a value of 0 as a byte
endTransaction	End the SPI transaction (should be called after a digitalWrite(pin_N, HIGH) is called on the CS line)	`SPI.endTransaction();`	End the SPI transaction

I²C communication principle

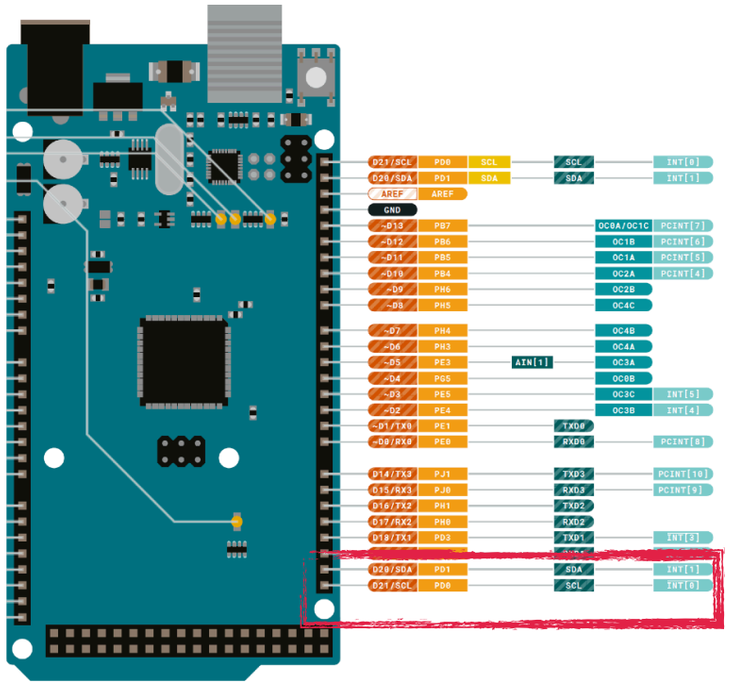
- Advantages compared to the other two communication methods:
  - Synchronous - e.g. tied to a clock signal
  - Ability to connect multiple controllers to multiple peripherals
  - Simplicity: implementation requires two lines + some resistors
- Two lines: a data line (SDA) and a clock line (SCL)
  - Both lines pulled high in their idle states. When data needs to be sent over, the lines are pulled low
  - Open drain system: lines can be only driven low by the devices
- Pull-up resistors (usually ~4.7 kΩ) to ensure that the line is pulled high in the idle state



No separate digital lines: peripherals are selected via unique **byte addresses**



I²C on Arduino



- Implementation on Arduino through the Wire.h library
- Multiple controllers can be connected over I2C by connecting SDA and SCL lines to the bus's lines.
- Only one controller can communicate at one time, and controllers can't communicate between each other over the same I²C bus.
- I²C is *half-duplex*, like UART

I²C (Wire) Method	Purpose	Example Code	Explanation
begin	Initiate the library and join the I²C bus as either a controller or peripheral	Wire.begin();	Joins the I²C bus as a controller. If an address is specified as a parameter to the method, the Arduino joins the bus as a peripheral with that address.
beginTransmission	For Arduino configured as a I²C controller: initiate transmission with the peripheral that has the given address	Wire.beginTransmission(0x68);	Begins transmission with the peripheral that has the hexadecimal address ox68
write	Write the byte data over the I²C bus	Wire.write(0x6B);	Writes byte data of value ox6B over the I²C bus
requestFrom	Request a specified number of bytes from the peripheral with the given address; Has the option of releasing the I²C line or holding onto it for further communication The requested bytes are put into a buffer to be subsequently read through *Wire.read()* calls.	Wire.requestFrom (0x68, 6, true);	Requests 6 bytes from the peripheral that has the address ox68. Releases the I²C line after the request is complete. If the last parameter is *false*, the Arduino will hold onto the I²C line for further communication, not allowing other devices to communicate over it.
read	Read bytes put into the buffer after transmissions from the peripheral device. This method is called *after* a call to *Wire.requestFrom*	Wire.read();	Reads one byte from the buffer (after a call to *requestFrom*) has been made. To read N bytes from the buffer, this method must be called N times
endTransmission	Ends the current transmission; Has the option of releasing the I²C line or holding onto it	Wire.endTransmission(true);	End transmission and releases the I²C line.

Time to get started!

- Checkout the Arduino IDE
- The Arduino has a built-in LED [guess what variable you can use to access it if you don't remember the pin # by heart... LED_BUILTIN].
- Find an example code that shows you how to blink the LED. Try to understand the code, ask the instructor or the TA if there are questions.
- Once understood how to blink the LED, let's have it blinking your initials (American name) using Morse Code

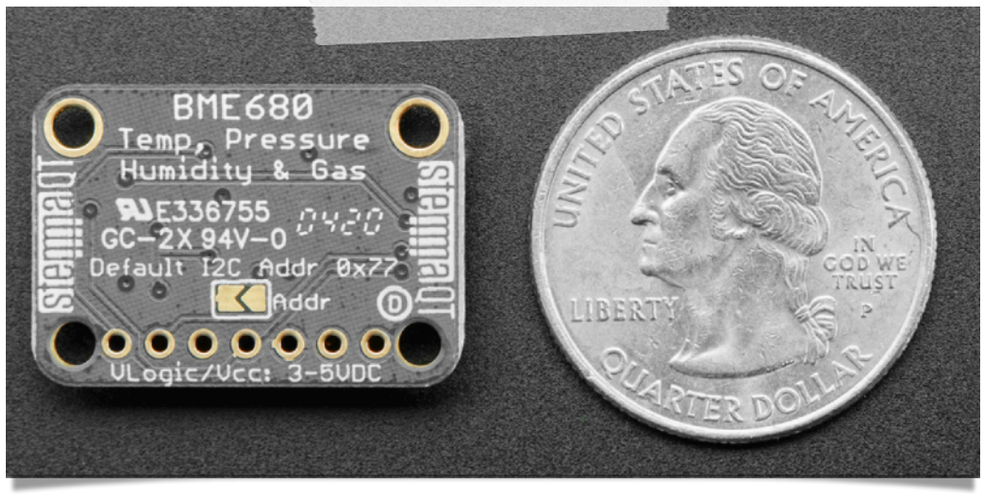
A •—	N —•	1 •————	? ••——••
B —•••	O ———	2 ••———	! —•—•——
C —•—•	P •——•	3 •••——	. •—•—•—
D —••	Q ——•—	4 ••••—	, ——••——
E •	R •—•	5 •••••	; —•—•—•
F ••—•	S •••	6 —••••	: ———•••
G ——•	T —	7 ——•••	+ •—•—•
H ••••	U ••—	8 ———••	- —••••—
I ••	V •••—	9 ————•	/ —••—•
J •———	W •——	0 —————	= —•••—
K —•—	X —••—		
L •—••	Y —•——		
M ——	Z ——••		

Morse Reminder:
- ➡ Dot = 1 second
- ➡ Line = 3 seconds
- ➡ Space between dots and lines in the same letter = 1 second
- ➡ Space between letters = 3 seconds
- ➡ Space between different words = 7 seconds

Milestone 1a... 29 to final grade!

Time to add our first sensor!

- Find the BME680 from your course-kit box



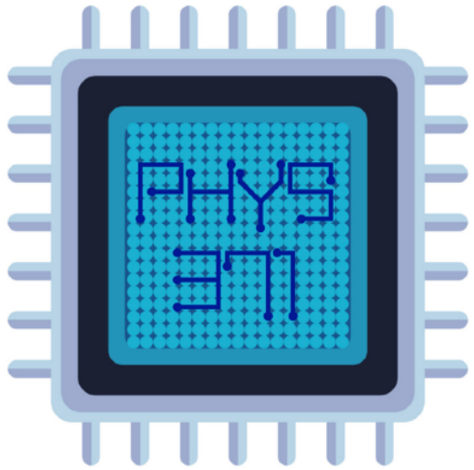
Adafruit Documentation

BME680 - “all the environmental sensing you want” in one small package.

More details next week, in one of your talks!

- Install it on your breadboard. How do you wire it? Try to check out the **schematics available on the course webpage** or browse the Adafruit documentation for the sensor

Milestone 1b… 28 to final grade!



HOMEWORK

Dr. Riccardo Longo

01/20/2023

Lecture 1

List of Sensors in your starter pack

In the kits we handed off to you - you can find:

- **TCA9548A I2C Multiplexer**
- **P1890 Mini Metal Speaker**
- **Keypad**
- **W12934-A MicroSD card breakout board**
- **LSM90S1 9 DOF**
- **Mic Amp MAX 4466**
- **Mini 2-wire Volt Meter**
- **MCP4725 DAC**
- **BME680**
- **Mono 2.5W Amp PAM8302A**
- **Ultimate GPS Breakout**
- **MLX90614 3V**
- **DS3231 Precision RTC**
- **DPS310 Pressure**

Short group talks next week

In the kits we handed off to you - you can find:

- TCA9548A I2C Multiplexer
- P1890 Mini Metal Speaker
- Keypad
- W12934-A MicroSD card breakout board
- LSM90S1 9 DOF
- Mic Amp MAX 4466
- Mini 2-wire Volt Meter
- MCP4725 DAC
- **BME680** —> short presentation (~10') by group X next week - volunteers?
- Mono 2.5W Amp PAM8302A
- Ultimate GPS Breakout
- MLX90614 3V
- **DS3231 Precision RTC** —> short presentation (~10') by group Y next week - volunteers?
- DPS310 Pressure

Homework assignment

https://courses.physics.illinois.edu/phys371/sp2023/homeworks.asp

Questions?

Week 1 homework

Due date and so forth

Please email your completed assignment to the course TAs (mch6@illinois.edu and jjc11@illinois.edu cc rlongo@illinois.edu, subject: '[PHYS371]: Week 1 Homework, Your Name') by Thursday, 5 pm of week 2 (01/26/2023). Each day of delay in turning in the assignment will result in a grade reduction of 10%. We will not grade anything submitted more than one week late.

When your homework submission includes one or more Arduino code files, please use the template **p398dlp_template.ino** as the starting point for your code. (I have it posted to the course homeworks web page.) Please fill in ***all*** of the fields shown in the template file.

In addition, your homework submissions—code, cell phone photos, etc. must include enough identifying information for us to tell who you are! Please compress all the material related to the homework in a .zip or .tar file.

Problem 1.

During the semester, we'll be working with the Arduino IDE, the Anaconda Python IDE, and the AutoDesk products Tinkercad and (perhaps) EAGLE PCB. Downloads and student registrations are free for all of these. You may have been able to download and install the two IDEs, and register for a (free) student account with AutoDesk. If not, do it now.

Anaconda Python has considerably better debugging and code development tools than Jupyter, with which you might have already worked, so I want you to install the Anaconda Python software on your laptop.

Please submit screenshots from your laptop of the Arduino and Anaconda Python IDEs, and also a Tinkercad.com page that shows that you have logged into your account, rather than just showing the free, no-account-needed page. For Tinkercad, please use the link https://www.tinkercad.com/joinclass/VP9YD3TIK to join the class repository (the username to be used is your NetID).

Please also create a group repository for your project code on GitLab (https://gitlab.engr.illinois.edu). Name the repository *Group-XX-PHYS371-SP2023* and add the instructor (rlongo) and the TAs (mch6 and jjc11).

Problem 2.

Add a BME680 and an LCD to your breadboard using the wiring indicated in the schematic linked to "GG's data logger schematic for spring 2021" on the course's "Code and design resource repository" web page. You should be able to find a demo code that'll let you read the temperature, pressure, and humidity from the BME680 and display them on the LCD. Have the display update about once per second.