PHYSICAL PROTOTYPING

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

AGENDA

Messages

Project planning and prototyping

Using prototypes in concept evaluations

Digital Fabrication

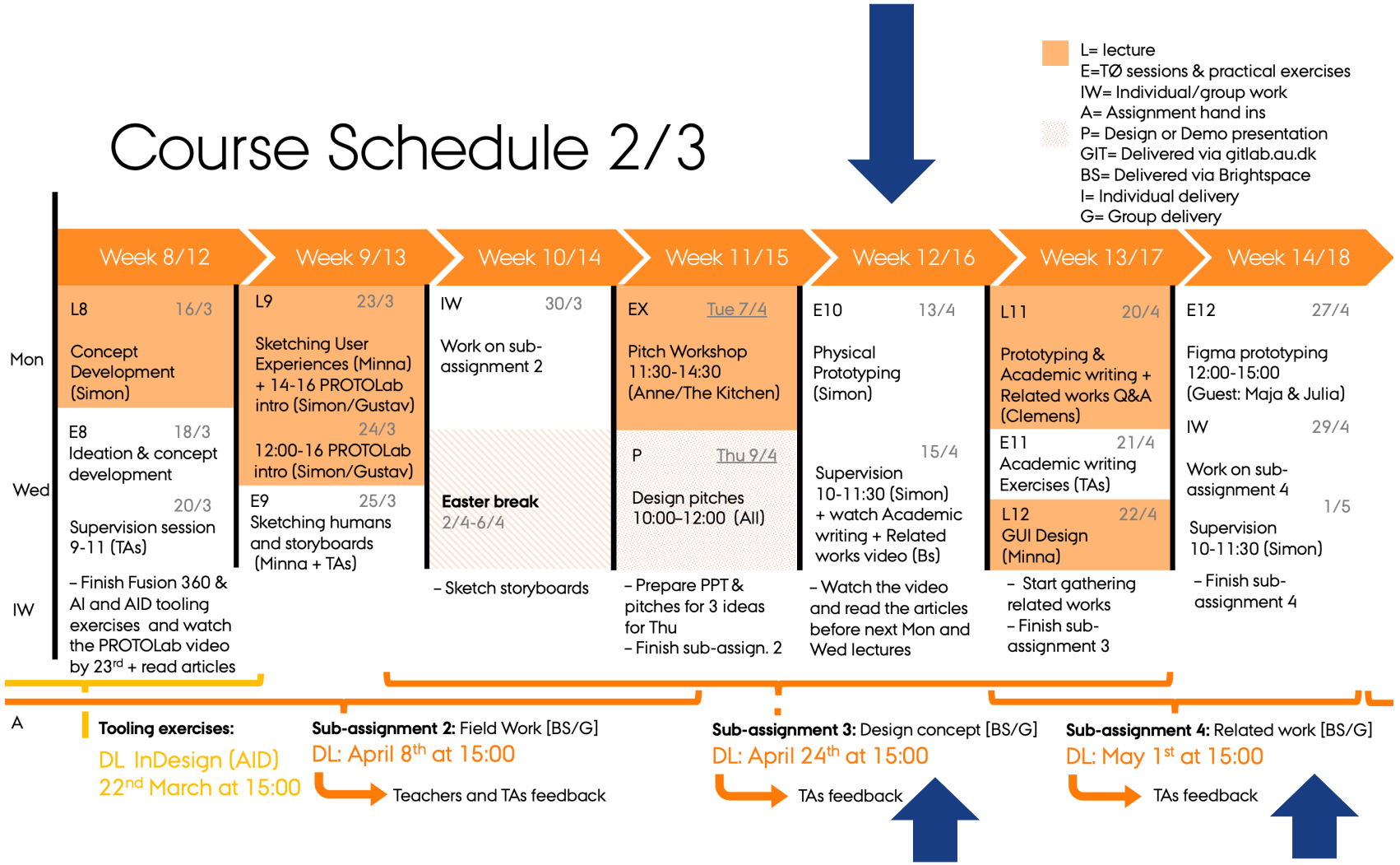
Electronics

Supervision after lecture

MESSAGES

THE ITPDP PROJECT

Course Schedule 2/3



THE ITPDP PROJECT

Course Schedule 3/3

L= lecture
E=TØ sessions & practical exercises
IW= Individual/group work
A= Assignment hand ins
P= Design or Demo presentation
GIT= Delivered via gitlab.au.dk
BS= Delivered via Brightspace
I= Individual delivery
G= Group delivery

26th Jun

	Week 15/19	Week 16/20	Week 17/21	Week 18/22	Week 19/23	Week 20/24	Week 22/26
Mon	P 4/5 Prototype Demo Day 12:00-15:00	IW 11/5 Conduct and report user evaluations	L14 18/5 Logo design (Minna)	L15 Tue 26/5 Poster design (Minna)	IW 1/6 Exam info (Minna / Simon)	P 11/6 1st iteration poster presentation & feedback session at 10-12 (Minna & TAs)	P 22/6 Final prototype exhibition with a product poster
Wed	L13 + E13 6/5 Evaluation (Minna) +Evaluation planning (TAs) 8/5 Supervision 10-11:30 (Simon)	IW 13/5 Conduct and report user evaluations	E14 20/5 Logo design exercises 22/5 Supervision 10-11:30 (Simon)	E15 27/5 Poster design exercises 29/5 Supervision 10-11:30 (Simon)	IW 3/6 Final report writing & demo video	IW Finalizing the prototype and poster for the final exhibition	Exams 23-25/6 9:15–> 26/6 Chomsky cleaning & course evaluation discussion
IW	– Write the design and implementation section –Prepare for the user evaluations	– Finish sub-assignment 5 by Tue	–Analyze user study results	– Finish sub-assignment 6 by Fri	– Finish the report by Thursday	– Finish the poster based on the feedback and upload it to Dropbox by 17th June at 12:00	– Everyone needs to be present in the demonstration, exam, & cleaning of the Chomsky & Feedback discussion.
A	**Sub-assignment 5:** Design & implementation [BS/G] DL: May 12th at 15:00 TAs feedback	**Sub-assignment 6:** Evaluation [BS/G] DL: May 29th at 15:00 TAs feedback			**Final report** [BS/G] DL: June 4th at 13:00	**Product poster** [BS/G] DL: June 17th at 12:00	

THE ITPDP PROJECT

From Brightspace under “ITPDP 2026”
Presented at first lecture about course introduction

Eligibility for exam

To be eligible for the exam, each student must hand in all tooling exercises and supporting assignments, and every group must hand in the final report, be represented in the plenary sessions, and - of course - the exam related demo session and oral exam.

Goals and requirements for the final project

The IT product must be placed clearly in relation to one of the three subthemes (or have deviation agreed upon with course educator).

The group must demonstrate an understanding of the target user group's needs and conditions, using theory and tools introduced in this course, and/or theory/tools presented earlier in their study programme.

The solution must position itself as relevant to the future users, utilizing ethnomethodological work and user-centered design to ensure validity. A successful evaluation is however not a requirement, but reflection and analysis of validity, process and outcome is the main end result.

The context of use and the challenges identified in the user studies must be taken into account.

The solution must include multiple clients/components and the cloud (typically via web technology). Several subcomponents and resources that can communicate, e.g., a system with a mobile app, a physical installation, and with automatic storage of sensor data in the cloud.

The IT product may involve a smartphone as part of the solution, but there must be an interaction with physical/tangible components in the environment or on the user, which you have designed and fabricated yourself.

Arguments must be made based on the needs of the users, usage of theory and methods, and the business potential in relation.

WORKLOAD AND EXPECTATIONS

European standard: 1 ECTS = 25-30 hours of work.

For Aarhus University, it has been set at 28 hours.

ITPDP is slightly different.

First part (seen as a 5 ECTS course): 140 hours per person, including lectures/TØ, from 26th of January to 17th of March.

Second part (seen as a 15 ECTS course): 420 hours per person, including lectures/TØ, from 18th of March to June 17th. This is equal to a minimum of 32 hours per week.

The rest of the course schedule is exam period and exam preparation time.

WORRYING TENDENCIES

Resource hour overview:

Estimated teaching time: 68-70 hours, not including supervision.

1 person group: The overall project should reflect 490 hours of work.
2 person group: The overall project should reflect 980 hours of work.
3 person group: The overall project should reflect 1470 hours of work.
4 person group: The overall project should reflect 1960 hours of work.

In the current state, your project should currently represent roughly 230 hours per person.
For a three person group = 425 hours of work (*30% deducted, prioritization*)

Can you honestly say that you have put this time into the project, preparation, lectures, reading etc.? If yes, then great!

PROJECT PLANNING

A university course.

Project planning and implementation is part of the learning outcomes – we cannot guide and/or hand-hold you the entire way.

Make sure to keep yourself updated via the Course Schedule on Brightspace.

Please ignore all other sources, as mentioned multiple times.

Be present… We cannot teach or supervise students who do not show up.

You can *technically* stay away from now until final report and exam, as long as Interview Assignment and Tooling Exercises are done. But don't.

PHYSICAL PROTOTYPING

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

MVP

“… Version of a product with just enough features to be usable by early customers”

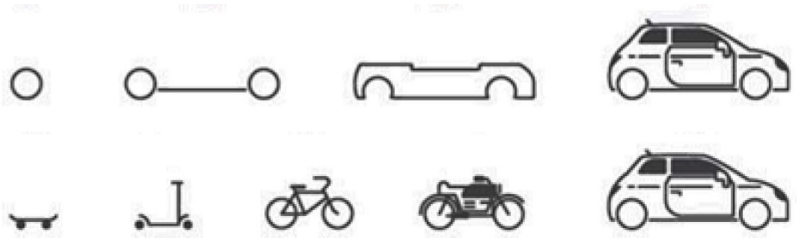
MINIMUM VIABLE PRODUCT (MVP)

Often a term used in “Agile Prototyping”.

A way to create clear goals to be able to evaluate/validate a prototype/concept.

What is the minimum features our prototype needs to have, in order to solve the identified problem and/or answer the research question?

Build towards simplistic representations of your vision – not individual small parts.



How not to build a *Minimum Viable Product*

How to build a *Minimum Viable Product*

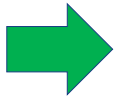
Agile Prototyping for technical systems – Schuh et al.

BUT HOW DO YOU PLAN THIS?

https://activecollab.com/blog/project-management/moscow-method

MoSCoW EXAMPLE - KITCHEN TIMER

But... how do we prioritize these??



Must have

- Time can be set (up to two hours)
- Hours and minutes are separate
- The user gets notified
- The timer can be reset
- Timer can be paused
- Fits in the kitchen
- Battery lasts more than max. timer

Should have

- Timer can be set (up to 24 hours)
- Hours, minutes and seconds
- User gets multimodal feedback
- Multiple concurrent timers
- Small enough to carry
- Battery lasts for full longitudinal study duration

Could have

- Timer = multiple weeks?
- Smartphone connected
- Contextual multimodal feedback

Will not have

- Cloud-connection
- Statistics and food categorization
- Credit-card like footprint
- Fusion sensing (thermometer?)
- Solar panel

HOW TO PRIORITIZE POST MoSCoW

DEPARTMENT OF COMPUTER SCIENCE
AARHUS UNIVERSITY

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

HOW TO PRIORITIZE IN MoSCoW

Aim high, scope appropriately – you should push yourself.

Free Features First

Impact-analysis (Remember scenarios? Personas? Contextual Design?)

The art of choosing the right ecosystem/backbone

Pessimistic approximations

AIM HIGH, SCOPE APPROPRIATELY

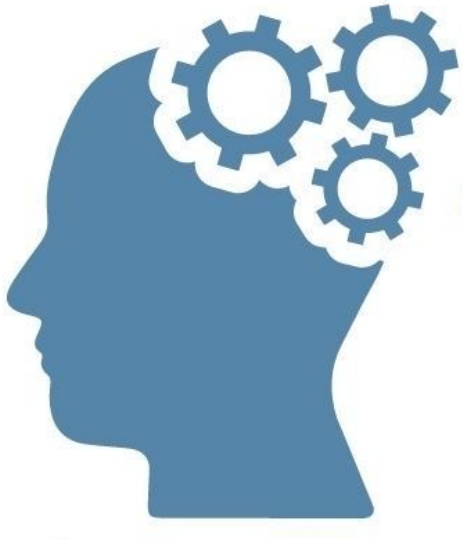
Push yourselves, but lean into your skillset.

Always scope based on project goals (in this case course goals and delimitations).

Do not let an overly simple MVP set you back.

An MVP is built to be your fallback – leave it alone, document, take pictures, make videos.

Your MVP is then baseline for next iteration.

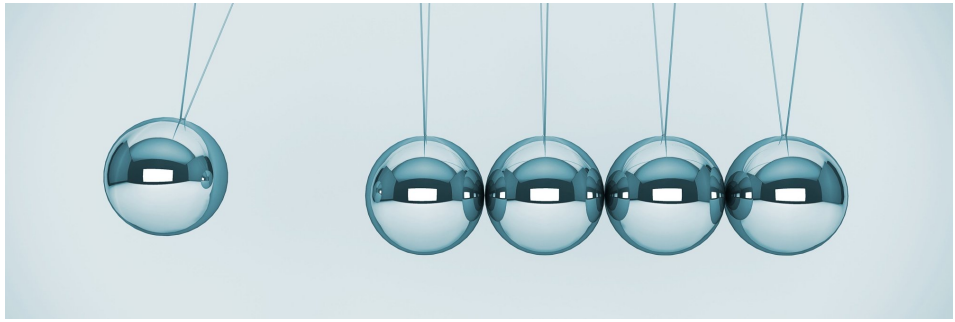


IMPACT ANALYSIS

Focus on scenario-based design and contextual design – use it!

Which features will have the most (positive) impact?

You understand your user and the context – use the analysis to strengthen arguments for feature prioritization.



IMPACT ANALYSIS - KITCHEN TIMER

Must have

- Time can be set (up to two hours) 1
- Hours and minutes are separate 1
- The user gets notified 2
- The timer can be reset 4
- Timer can be paused 5
- Fits in the kitchen 6
- Battery lasts more than max. timer 3

Should have

- Timer can be set (up to 24 hours) 7
- Hours, minutes and seconds 8
- User gets multimodal feedback 9
- Multiple concurrent timers 12
- Small enough to carry 11
- Battery lasts for full longitudinal study duration 10

Could have

- Timer = multiple weeks?
- Smartphone connected
- Contextual multimodal feedback

Will not have

- Cloud-connection
- Statistics and food categorization
- Credit-card like footprint
- Fusion sensing (thermometer?)
- Solar panel

“FREE FEATURES FIRST”

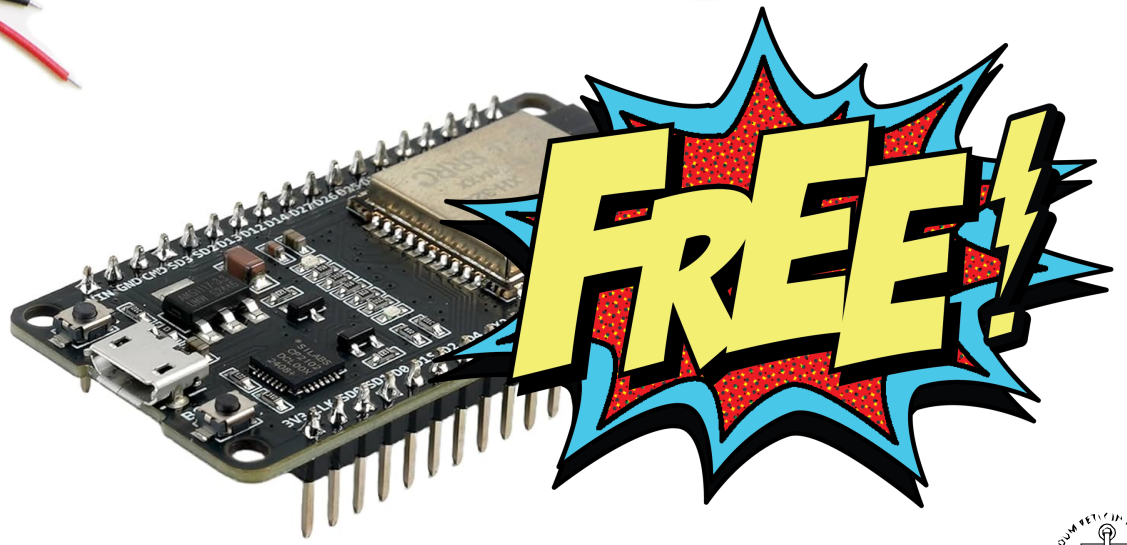
Free features are not necessarily free, but requires very little technical reiteration.

Hardware vs. software

Simple additions or changes

VS

VS



IMPACT ANALYSIS - KITCHEN TIMER

Must have

- Time can be set (up to two hours) 1
- Hours and minutes are separate 1
- The user gets notified 2
- The timer can be reset 4
- Timer can be paused 5
- Fits in the kitchen 6
- Battery lasts more than max. timer 3

Should have

- Timer can be set (up to 24 hours) 7 FREE!
- Hours, minutes and seconds 8
- User gets multimodal feedback 9
- Multiple concurrent timers 12
- Small enough to carry 11
- Battery lasts for full longitudinal study duration 10 FREE!

Could have

- Timer = multiple weeks? FREE!
- Smartphone connected FREE!
- Contextual multimodal feedback

Will not have

- Cloud-connection
- Statistics and food categorization
- Credit-card like footprint
- Fusion sensing (thermometer?)
- Solar panel

TIME AND SPRINTS

https://www.uxpin.com/studio/blog/design-sprints/

Agile development methodologies often refer to the term “sprint”.

I strongly suggest using this approach for testing feasibility in feature-based prototyping.

Suggested framework for prototype sprint:
Inspired by the often used “5-day design sprint” (or similar artzy naming scheme)

Following MoSCoW prioritization and free-features-first analysis.

Have initial brainstorm about implementation strategy.

Divide most crucial features up, and sprint them in pairs.

Work 24-72 hours on each feature depending on impact (two features being worked on simultaneously).

End with feasibility assessment and construct time plan based on last sprint. Or scrap.

PROTOTYPE EVALUATION

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

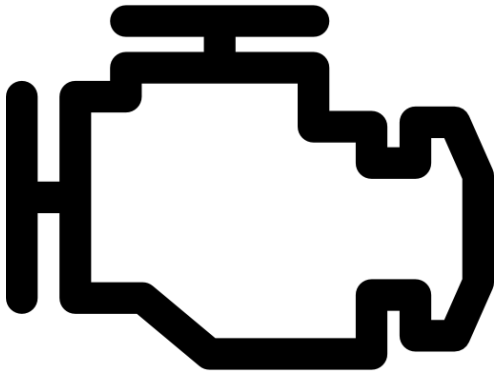
THE FALLACY OF FLAWLESS

Prototypes are inherently flawed.

Small factors and variance makes a difference for replicability and comparability.

The obvious end-goal is a self-sustained and robust system.

Often that is not the case...
and maybe not the best and easiest way to add additional features?



WIZARD OF OZ

https://www.nngroup.com/articles/wizard-of-oz/

FINISH, FIDELITY, FEEDBACK

Maryam Tohidi, William Buxton, Ronald Baecker, and Abigail Sellen. 2006. Getting the right design and the design right.

and

Youn-Kyung Lim, Erik Stolterman, and Josh Tenenberg. 2008. The anatomy of prototypes: Prototypes as filters, prototypes as manifestations of design ideas.

End goal of a prototype evaluation:

- To validate concept
- Evaluate user appropriation
- Problem-solution fit
- Feedback on future iterations

Prototype fidelity matters!

Users find it easier to critique lower fidelity prototypes.

Sometimes using multiple low-fidelity, feature-focused prototypes is the right way…

… For evaluation ☺

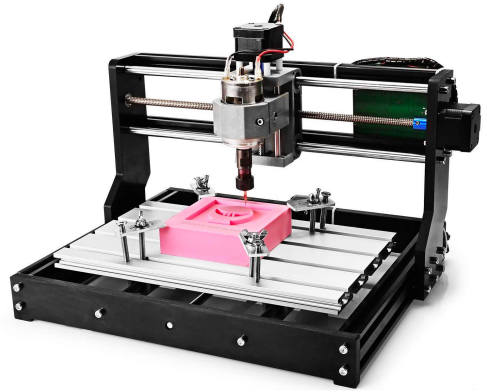
DIGITAL FABRICATION

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

MANUFACTURING METHODS

Additive Manufacturing	Subtractive Manufacturing
3D printing as example	CNC (Computer Numerical Control) milling as example
Adding layer by layer	Taking away layer by layer
Material base = often plastic	Material base = often metals/wood
Often used for fast initial prototyping	Often used for sturdy construction

Combo =
Hybrid Process



Meh?

1. What type of features does your product have?
   - small organic and intricate features → additive methods
   - large or sharp features, drilled and tapped holes or other fastening features →subtractive methods

2. **What type of material do you want to work with?**
   - **thermoplastics and resins → additive methods**
   - **materials like metals, wood, or foam → subtractive methods**

3. How many units do you want to produce?
   - low-volume production or iterative prototyping → additive methods
   - large-volume production runs → subtractive methods

Meh?

RAPID PROTOTYPING

Purpose: Rapidly creating tangible artifacts or prototypes to filter/validate certain aspects.

Addendum: In house. By yourself. Something you hopefully made. Without ruining yourself…

… or the equipment ☺

Most common practice and understanding:

Utilizes digital fabrication, due to speed and work/process synchronizity.

Any fabrication occupying more than a few days is no longer rapid prototyping – it's invested construction.

Invested construction is not a bad thing – perfect for prototyping towards final demo.

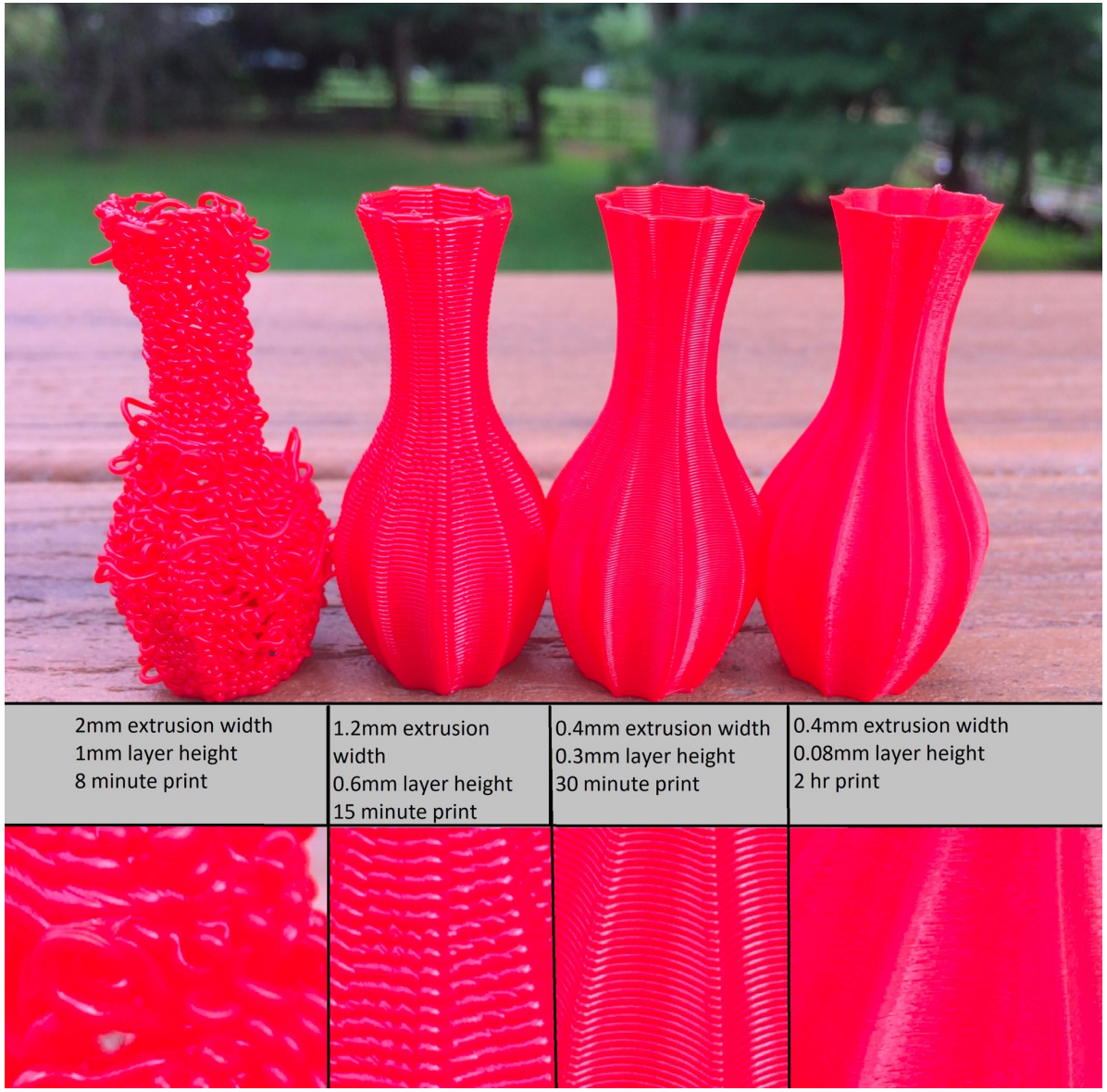
Rapid prototyping is your tool to quickly test, validate and scrap. Anything too detailed - and invested - in can become hard to scrap.

RAPID PROTOTYPING

Mastering techniques, process, equipment

=

<u>Absolute control regarding time and scope</u>



ARDUINO

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

AGENDA

- Why Arduino for prototyping
- Alternative prototyping platforms (and what you will be taught about in other courses)
- Code examples
- ESP32 Wizard of Oz

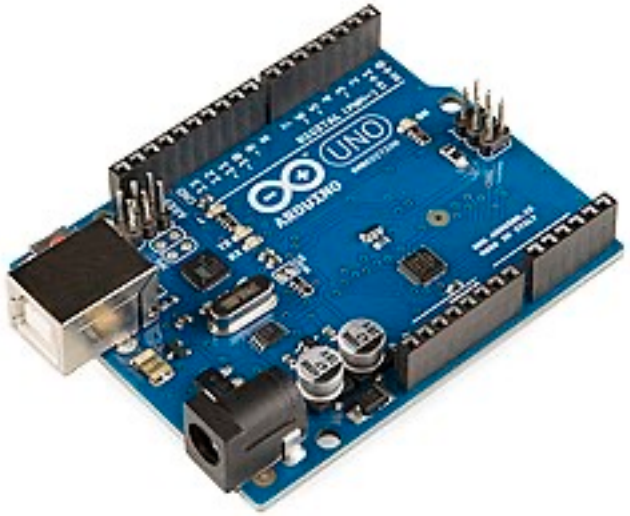
ARDUINO AS PROTOTYPING PLATFORM

Readily available here and big part of our lab-ecosystem

Lots of documentation and code examples + tutorials

Easy prototyping, hook-up, HATs, powering etc.

Many variations, sizes and beefiness



Often limiting: CPU/GPU power, threading/cores, memory, GPIO pin amount, buffer sizes, communication.

ALTERNATIVE PLATFORMS

Wemos D1 Mini (cheap WiFi board based on ESP8266)

ESP 32 (great platform with lots of possibilities)

RPI (when you want something in between a PC and an Arduino)

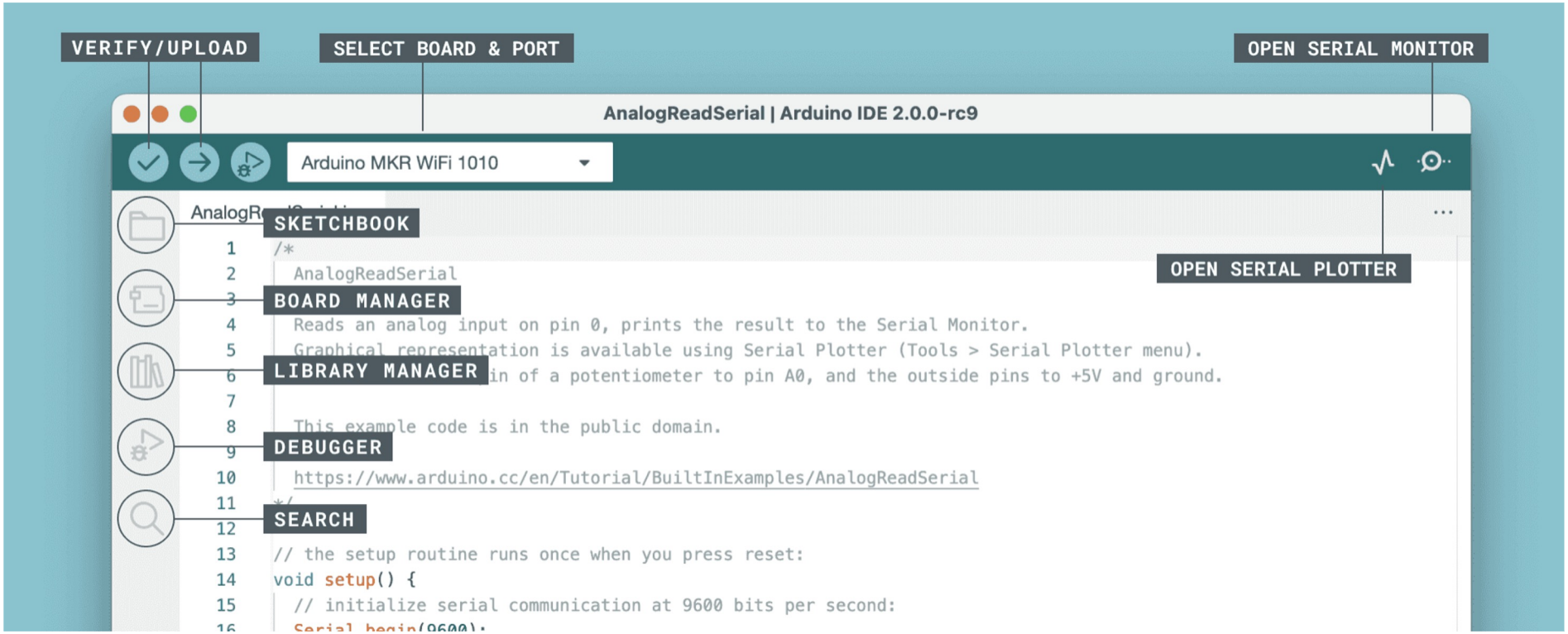
ATMEL MCU family (used in PhysComp. When you want to create custom circuits)

NUCs/Smartphones (for webapps/apps, graphics-intense tasks)

ARDUINO IDE

Arduino is an open-source electronics platform based on easy-to-use hardware and software.

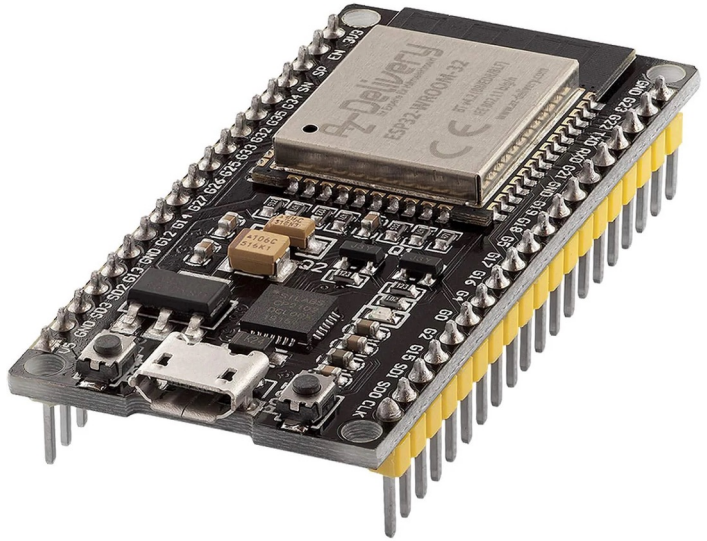
https://www.arduino.cc/en/software



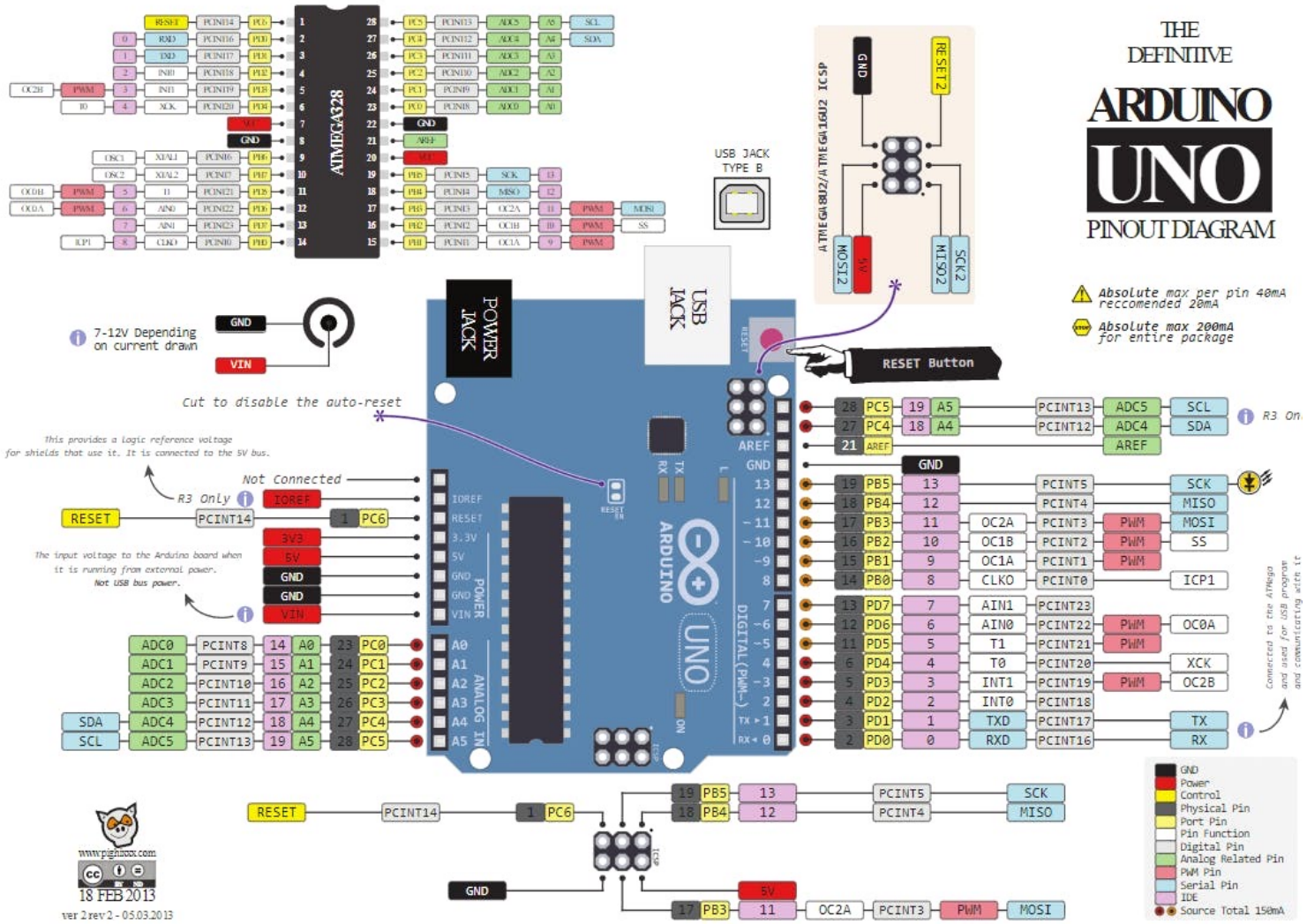
ESP32

ESP32 is widely used in System-on-Chip (SoC) solutions designed for Internet of Things (IoT) applications due to its:

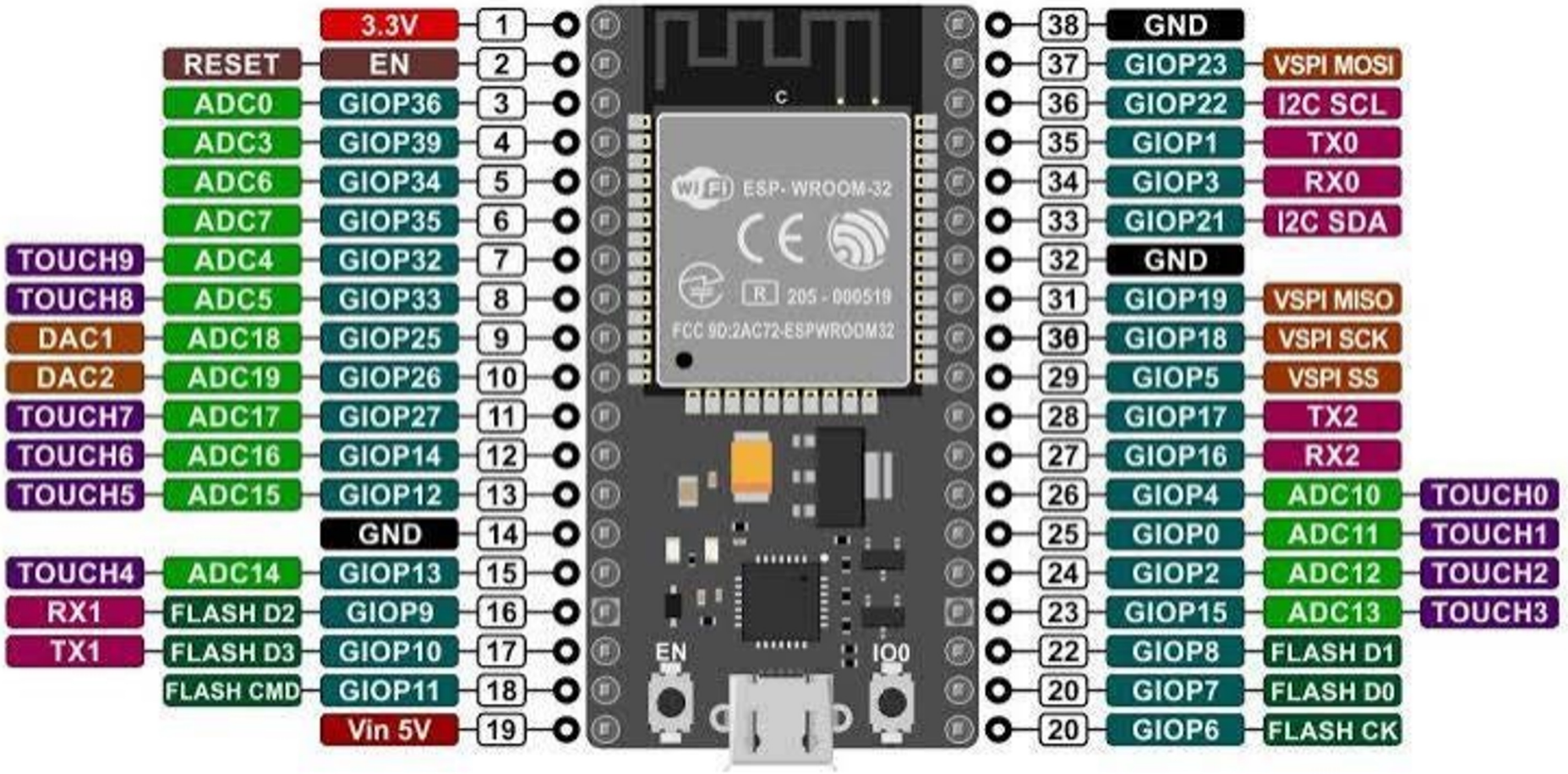
- Integrated WiFi and Bluetooth
- Lower power consumption with deep sleep modes
- More I/O Features (than Arduino)



PINOUTS?


THE DEFINITIVE
ARDUINO
UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK TYPE B
POWER JACK
USB JACK
RESET Button
7-12V Depending on current drawn
Cut to disable the auto-reset
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
Not Connected
R3 Only
RESET
3V3
5V
GND
VIN
AREF
SCL
SDA
SCK
MISO
MOSI
SS
ICP1
OC0A
XCK
OC2B
TX
RX
ATMEGA8U2/ATMEGA16U2 ICSP
www.pighixxx.com
18 FEB 2013
ver 2 rev 2 - 05.03.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

ESP32


3.3V 1
RESET EN 2
ADC0 GIOP36 3
ADC3 GIOP39 4
ADC6 GIOP34 5
ADC7 GIOP35 6
TOUCH9 ADC4 GIOP32 7
TOUCH8 ADC5 GIOP33 8
DAC1 ADC18 GIOP25 9
DAC2 ADC19 GIOP26 10
TOUCH7 ADC17 GIOP27 11
TOUCH6 ADC16 GIOP14 12
TOUCH5 ADC15 GIOP12 13
GND 14
TOUCH4 ADC14 GIOP13 15
RX1 FLASH D2 GIOP9 16
TX1 FLASH D3 GIOP10 17
FLASH CMD GIOP11 18
Vin 5V 19
38 GND
37 GIOP23 VSPI MOSI
36 GIOP22 I2C SCL
35 GIOP1 TX0
34 GIOP3 RX0
33 GIOP21 I2C SDA
32 GND
31 GIOP19 VSPI MISO
30 GIOP18 VSPI SCK
29 GIOP5 VSPI SS
28 GIOP17 TX2
27 GIOP16 RX2
26 GIOP4 ADC10 TOUCH0
25 GIOP0 ADC11 TOUCH1
24 GIOP2 ADC12 TOUCH2
23 GIOP15 ADC13 TOUCH3
22 GIOP8 FLASH D1
20 GIOP7 FLASH D0
20 GIOP6 FLASH CK
ESP-WROOM-32
205 - 000519
FCC ID:2AC72-ESPWROOM32
EN
IO0

ARDUINO V.S. ESP32 V.S. ESP8266

Feature	Arduino	ESP32	ESP8266
CPU	8-bit, 16 MHz	Dual-core, 240 MHz	Single-core, 80–160 MHz
Memory	Low (2 KB SRAM)	High (520 KB SRAM)	Moderate (160 KB SRAM)
Wi-Fi	No	Yes (Wi-Fi + Bluetooth)	Yes (Wi-Fi only)
GPIOs	~14	~34	~17
Power	High consumption	Low-power modes (~5µA)	Deep sleep (~10µA)
Analog	10-bit ADC	12-bit ADC + 2 DACs	10-bit ADC
Security	Basic	Hardware encryption	Basic
Best For	Simple projects	IoT, AI, wireless apps	IoT with Wi-Fi

ARDUINO COMPONENTS

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

COMPONENTS – WHAT OPTIONS DO WE HAVE?

- Many components will be covered in Physical Computing next semester:
    - Entire lecture set on sensors and what types we have in lab as inspiration. Engineering Interactive Technologies delves further into making novel sensing techniques.
    - Entire lecture set on actuators – even more time to delve into actuators in Multimodal Interaction and Shape Changing Objects and Spaces
- **For now: Use Johannes' component inspiration kit and chomskylab.dk – and talk to supervisors/TAs/Labtools for further advice/inspiration.**

ARDUINO CODE EXAMPLES

SIMON HOGGAN CHRISTENSEN
LAB COORDINATOR

ARDUINO CODE EXAMPLES

“AnalogInOutSerial.ino”

“Arrays.ino”

”Smoothing.ino”

“switchCase.ino”

WiFi network examples (third-party examples)

ANALOGIN OUTSERIAL

Example of map

Example of analogRead

Example of Serial.print

```
// 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 = 9;  // 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);
}
```

ARRAYS

Array example because slightly different syntax compared to java

```
int timer = 100;  // The higher the number, the slower the timing.
int ledPins[] = {
  2, 7, 4, 6, 5, 3
};                // an array of pin numbers to which LEDs are attached
int pinCount = 6;  // the number of pins (i.e. the length of the array)

void setup() {
  // the array elements are numbered from 0 to (pinCount - 1).
  // use a for loop to initialize each pin as an output:
  for (int thisPin = 0; thisPin < pinCount; thisPin++) {
    pinMode(ledPins[thisPin], OUTPUT);
  }
}

void loop() {
  // loop from the lowest pin to the highest:
  for (int thisPin = 0; thisPin < pinCount; thisPin++) {
    // turn the pin on:
    digitalWrite(ledPins[thisPin], HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(ledPins[thisPin], LOW);
  }

  // loop from the highest pin to the lowest:
  for (int thisPin = pinCount - 1; thisPin >= 0; thisPin--) {
    // turn the pin on:
    digitalWrite(ledPins[thisPin], HIGH);
    delay(timer);
    // turn the pin off:
    digitalWrite(ledPins[thisPin], LOW);
  }
}
```

RUNNING AVG

”Smoothing.ino” in Arduino--> Examples

Running average data smoothing

Useful for sensor and input with noise

Noise = variance in data without change in input

```
const int numReadings = 10;

int readings[numReadings];  // the readings from the analog input
int readIndex = 0;          // the index of the current reading
int total = 0;              // the running total
int average = 0;            // the average

int inputPin = A0;

void setup() {
  // initialize serial communication with computer:
  Serial.begin(9600);
  // initialize all the readings to 0:
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
    readings[thisReading] = 0;
  }
}

void loop() {
  // subtract the last reading:
  total = total - readings[readIndex];
  // read from the sensor:
  readings[readIndex] = analogRead(inputPin);
  // add the reading to the total:
  total = total + readings[readIndex];
  // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array...
  if (readIndex >= numReadings) {
    // ...wrap around to the beginning:
    readIndex = 0;
  }

  // calculate the average:
  average = total / numReadings;
  // send it to the computer as ASCII digits
  Serial.println(average);
  delay(1);  // delay in between reads for stability
}
```

SWITCHCASE

Crucial for ITTT cases and sensing

```
const int sensorMin = 0;    // sensor minimum, discovered through experiment
const int sensorMax = 600;  // sensor maximum, discovered through experiment

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
}

void loop() {
  // read the sensor:
  int sensorReading = analogRead(A0);
  // map the sensor range to a range of four options:
  int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

  // do something different depending on the range value:
  switch (range) {
    case 0:  // your hand is on the sensor
      Serial.println("dark");
      break;
    case 1:  // your hand is close to the sensor
      Serial.println("dim");
      break;
    case 2:  // your hand is a few inches from the sensor
      Serial.println("medium");
      break;
    case 3:  // your hand is nowhere near the sensor
      Serial.println("bright");
      break;
  }
  delay(1);  // delay in between reads for stability
}
```

ESP32 EXAMPLE – YOUR WOZ CHEATSHEET

ESP32 is a brilliant platform for the Wizard of Oz method.

Example code in Brightspace.

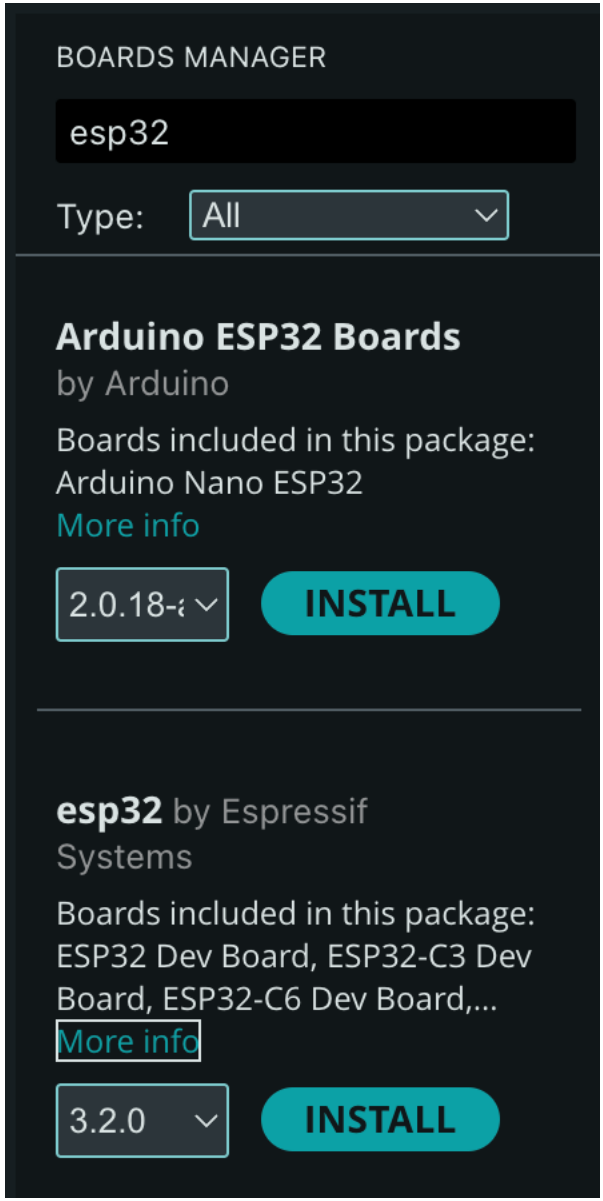
Simple UI via WiFiServer → interaction with ESP32

Easily adaptable.

Of course only used for evaluation and prototyping, not the final demo ☺

INSTALL THE ESP32 BOARD IN ARDUINO IDE

- Open IDE and go to **Tools > Board > Boards Manager**
- Search "ESP32", select "**esp32 by Espressif Systems**", and click Install
- Reopen Arduino IDE
- Under **Tools>Board** you should see **esp32 → ESP32 Dev Module option**



INSTALL USB CABLE DRIVER

Download Link: https://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers

- **Windows Users:** The Arduino Installer should install the USB Driver automatically. But in case it doesn't work, download from the link. Windows 10 users should use **CP210x Universival Windows Driver**, and Windows 8.1,8 and 7 users can use **CP210x Windows Driver.**
- **MacOS Users:** Download the **CP210x VCP Mac OSX Driver**. After installing, accept Privacy changes and prompts. If install fails, install legacy driver variant.
- **Testing** if the USB driver works (all OS): in the Arduino menu **Tools>port** *before* plugging in your ESP with the USB cable. You should see either none or one entry listed. Now **plug in your ESP microcontroller** and take another look at **Tools>port**. You should hopefully see one more entry. On Windows this will often be something like **COM3**. On Mac it will be **USB_to_UART or cu.usbserial**

TESTING AND GETTING INFO

- Open File→Examples → Basics → BareMinimum and test that upload works to your ESP32.

- During upload the device MAC adress will be written (including other useful info)

```
Connecting.....
Chip is ESP32-D0WD-V3 (revision v3.1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 3c:8a:1f:0c:64:00
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 921600
```

- Now it's time to get on WiFi, by either using your phone hotspot as WiFi, or connect to AU-IoT via Simon (bring him your device MAC address)

GETTING ON WIFI

```
#include <WiFi.h>

const char* ssid = "AU-IoT";
const char* password = "test1234";

void setup(){
Serial.begin(921600);
delay(1000);

WiFi.mode(WIFI_STA); //Optional
WiFi.begin(ssid, password);
Serial.println("\nConnecting");

while(WiFi.status() != WL_CONNECTED){
Serial.print(".");
delay(100);
}

Serial.println("\nConnected to the WiFi network");
Serial.print("Local ESP32 IP: ");
Serial.println(WiFi.localIP());
}

void loop(){}
```

Or your phone hotspot SSID + pass

GETTING ON WIFI

- Result 1:

- Result 2: Endless connecting loop (retry SSID/pass or another WiFi)
- Result 3: Connection failed (retry)
- Result 4: Gibberish (check baud rate).

REACTING TO ONLINE INPUT

- Open the .ino file from Brightspace called ”ESP32_WifiWorkshopFinal” (unedited code can also be found via link, remember to change SSID credentials and baud).

- Read through to understand what it does. It is a very basic HTTP-request ”ESP32 as web server”-setup, and is great for future prototyping!

- Set up a breadboard with your ESP and two LEDs to pin 26 and 27 (remember a 220 ohm resistor to each + GND pin)

- Use your computer/phone to control the pins (write the IP in the URL-field in a browser)

- **Optional - IoT Doorbell:** Edit the code to work with a single buzzer instead of two LEDs – maybe even make a little melody?

Idea and code from: https://randomnerdtutorials.com/esp32-web-server-arduino-ide/