Programming Computers

Oliver Sturm • @olivers • oliver@oliversturm.com

Computers Are Everywhere

- Obvious: PC, Mac
- Mobile: laptops, tablets, phones
- Small: watches, wearables
- Integrated/embedded: controllers in cars, radios, home electronics, USB sticks
- Sometimes the term "computer" is interpreted to include components other than the processor: memory, I/O chips, storage

Computers Are Everywhere

- Obvious: PC, Mac
- Mobile: laptops, tablets, phones
- Small: watches, wearables
- Integrated/embedded: controllers in cars, radios, home electronics, USB sticks
- Sometimes the term "computer" is interpreted to include components other than the processor: memory, I/O chips, storage

All processors require program code!

Program Delivery

- Desktop: installers
- Mobile: app stores
- Web: download in browser
- Controllers: factory preload, and/or developer upload through interfaces

Programming Languages - Assembler

Assembler "keywords" are translated (almost) directly into processor opcodes

```
iNumVec EQU [esp + 16]
pResult EQU [esp + 20]

mov ecx,iNumVec
mov eax,pMatrix
lea edx,[ecx*8]
neg ecx
```

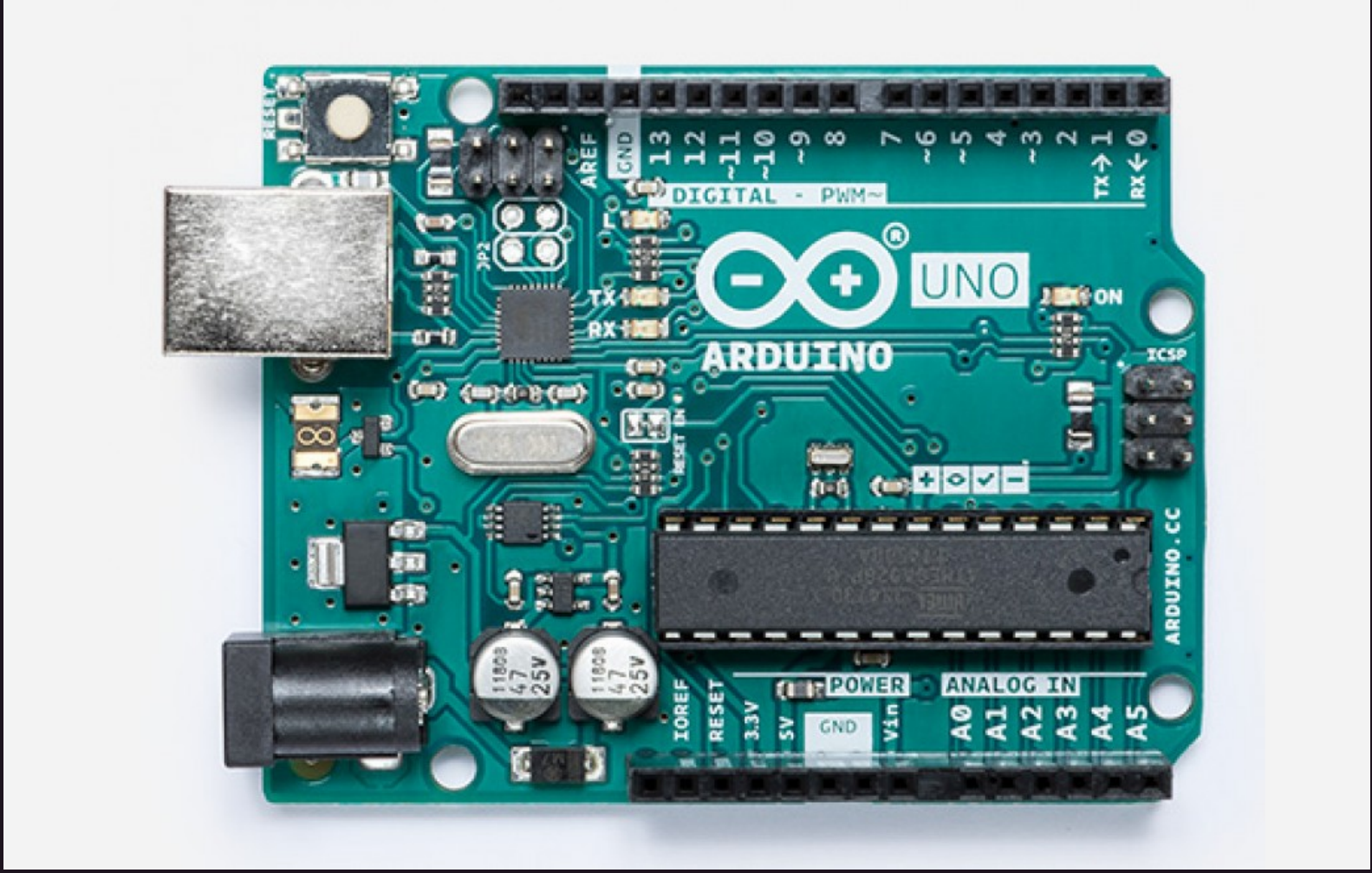
Programming Language Babylon

- Wikipedia lists 694 "notable programming languages"
- 694!
- Many languages are "special purpose"
- Many general-purpose languages have evolved and replaced one another
- StackOverflow survey 2019 still shows 11 languages used by at least 20% of the survey audience

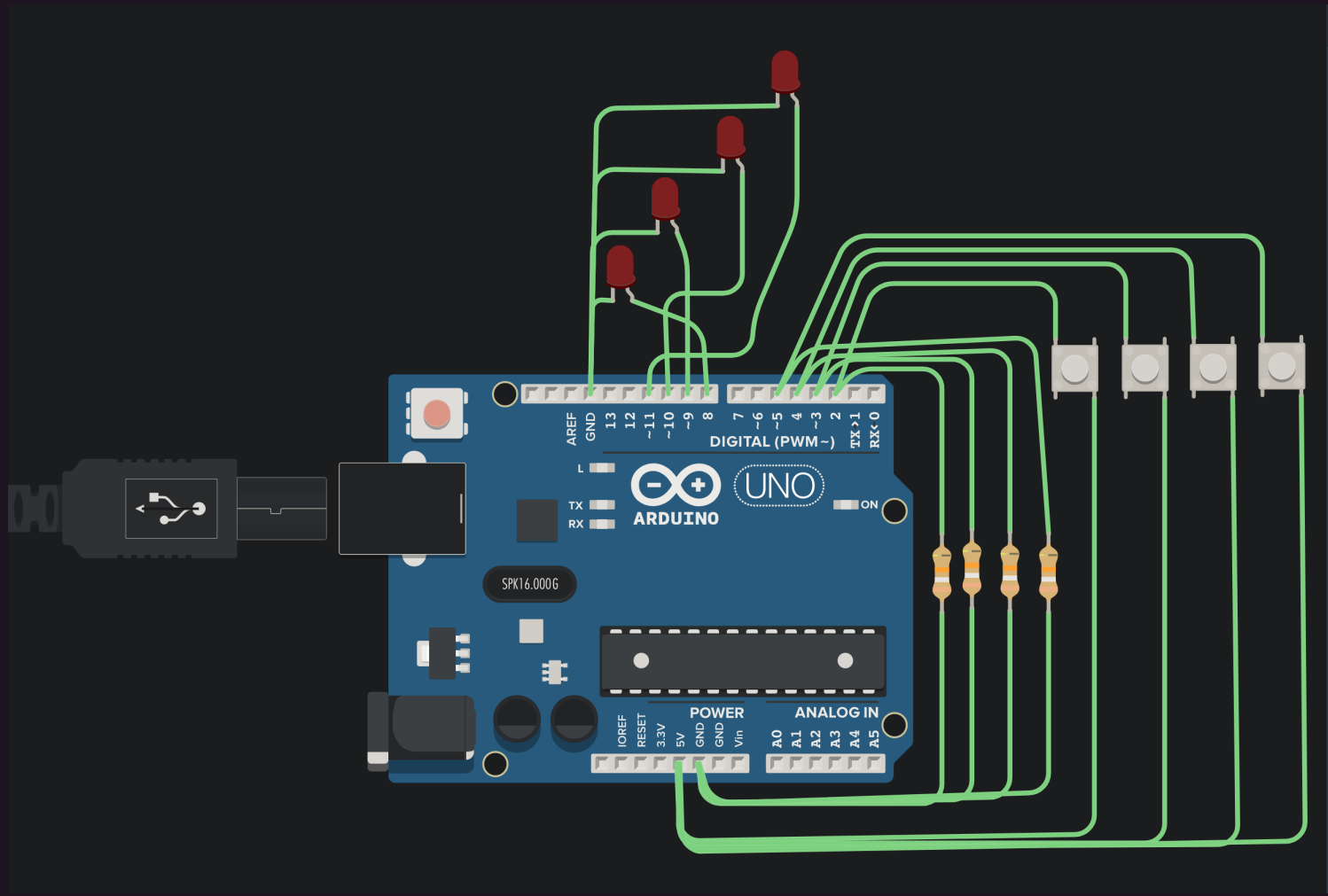
Programmable Controllers: Arduino

- https://arduino.cc
- Arduino IDE (Integrated Development Environment) free to download
- Lots of tutorials
- Basic principle of programming: IPO
  - Input / process / output

Programmable Controllers: Arduino



Arduino Sample - TinkerCad



Text

1 (Arduino Uno R3)

```
const int buttonsHandled = 4;
const int buttonIndexBase = 2;
const int outputIndexBase = 8;

int outputStates[buttonsHandled] = {LOW,LOW,LOW,LOW};
int buttonStates[buttonsHandled] = {LOW,LOW,LOW,LOW};

int lastReadButtonStates[buttonsHandled] = {LOW,LOW,LOW,LOW};


unsigned long debounceStartTime[buttonsHandled] = {0,0,0,0};
unsigned long debounceDelay = 100;    // the debounce time; increa

void setup() {
  for (int i = 0; i < buttonsHandled; i++) {
    pinMode(buttonIndexBase + i, INPUT);
    pinMode(outputIndexBase + i, OUTPUT);
    digitalWrite(outputIndexBase + i, outputStates[i]);
  }
}

void loop() {
  int readings[buttonsHandled];

  for (int i = 0; i < buttonsHandled; i++) {
    // read all the button states
    readings[i] = digitalRead(buttonIndexBase + i);

    // if the reading for this button has changed since last time,
    // remember the current time stamp as the starting point of a
    // debouncing timeframe
    if (readings[i] != lastReadButtonStates[i]) {
      debounceStartTime[i] = millis();
      lastReadButtonStates[i] = readings[i];
    }

    // if the last time we started a debouncing timeframe for this
    // button is longer than debounceDelay ago, consider accepting
    // a new state - this happens if no state change for the buttc
    // has been recorded for longer than debounceDelay
    if ((millis() - debounceStartTime[i]) > debounceDelay) {
      // see if the current reading actually differs from the prev
      // accepted state of this button
      if (readings[i] != buttonStates[i]) {
        // okay, we have a new buttonState - accept it
        buttonStates[i] = readings[i];

        // finally, if the accepted state is HIGH, toggle the outp
        if (buttonStates[i] == HIGH) {
          outputStates[i] = !outputStates[i];
          digitalWrite(outputIndexBase + i, outputStates[i]);
        }
      }
    }
  }
}
```

A Look At The Code (Main Loop IPO, cut to size)

```
void loop() {
  int readings[buttonsHandled];

  for (int i = 0; i < buttonsHandled; i++) {
    // Input: Read all the button states
    readings[i] = digitalRead(buttonIndexBase + i);

    // Process: find out whether output should be switched
    if ((millis() - debounceStartTime[i]) > debounceDelay) {
      // Output: set the new state
      digitalWrite(outputIndexBase + i, outputStates[i]);
    }
  }
}
```

Other Small Programming Platforms

- Microbit (https://microbit.org)
  - Great for learning
  - Visual Programming
  - JavaScript
- Raspberry Pi
  - Complete computer, but cheap and small
  - Linux, Windows (IoT Core), and others
  - Automation, network systems, entertainment
- PocketChip
  - All-in-one mini computer

Writing "real" computer applications

- User interfaces required
  - View description languages
- Native vs cross-platform
- Most popular application platform now: HTML/CSS/JS
- Learn in sandbox environments, e.g. https://codepen.io

Sample HTML

```
<p>
  <button onclick="button_clicked()">Do something</button>
</p>

<p>
  <input type="text" id="name" placeholder="Please enter your name" />

  <button onclick="sayhi()">
    Click here once you've entered your name
  </button>
</p>

<p id="output"></p>
```

Sample CSS

```css
button:hover {
  background-color: red;
}
```

Sample JavaScript

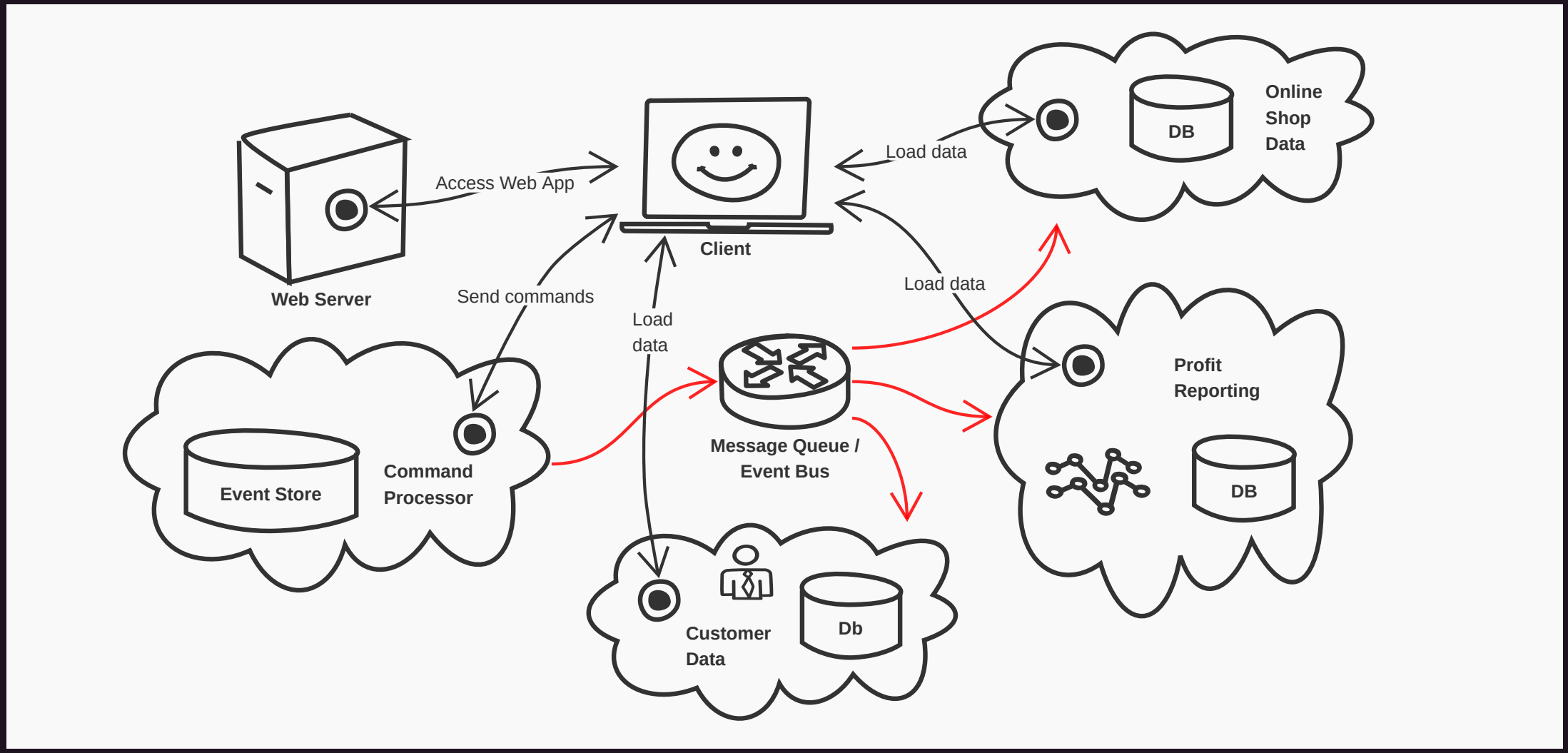
```
function button_clicked() {
  alert('Thanks for clicking the button');
}

function sayhi() {
  const nameField = document.getElementById('name');
  const outputField = document.getElementById('output');
  outputField.innerHTML =
    'You said your name is <b>' + nameField.value + '</b>';
}
```

Frameworks And Libraries

- Programmers are lazy
- Code can be distributed and reused
- Examples
  - UI libraries
  - Network functionality
  - Working with persistent data
  - Science and math functionality
  - Hardware interaction

Software Architecture



Questions?

Thank You

Please feel free to contact me about the content anytime.

Oliver Sturm • @olivers • oliver@oliversturm.com