UNIVERSITY OF TWENTE.

APPLICATION DEVELOPMENT

LECTURE 4: ARDUINO PROGRAMMING, C++, LISTS

```
class AppDev {
```


```
}
```

Part of SmartProducts

INTRODUCTION

APPLICATION DEVELOPMENT

- Arduino programming in C++

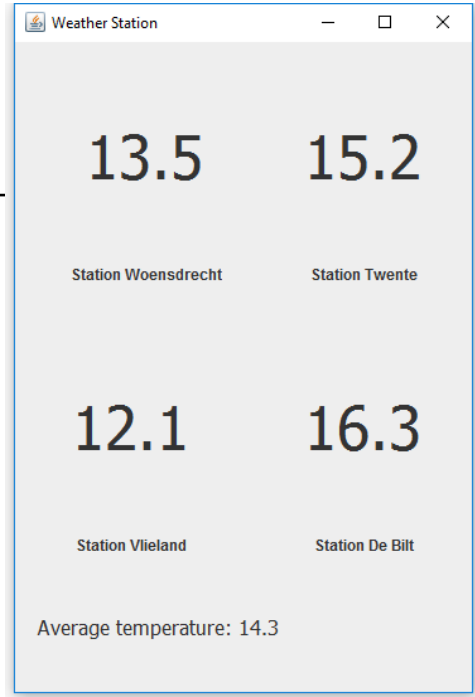
  (comparison with Java)
- Lists (Arrays)
- Assignment

Fjodor van Slooten
W241 *(Horst-wing West)*
f.vanslooten@utwente.nl

There are 3 ways to communicate with Fjodor: email, website chat, forum
(so no Canvas messages please, I will never see them)

slides @ vanslooten.com/appdev

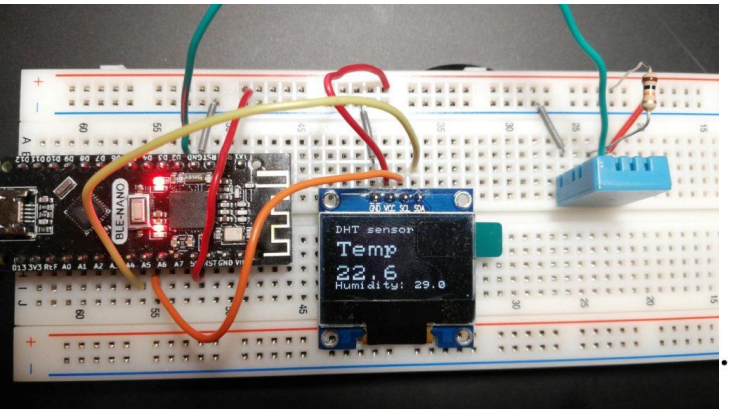
ASSIGNMENTS



- Java assignment most of you did very well

Practical:

- Build Arduino circuit which displays temperature & humidity
- Connect to phone: some of you had difficulties



- Use website chat/forum if you need help!

ABOUT THE PRACTICALS

- Some results…

DHT_Unified_Sensor_BLE_chat | Arduino 1.8.12

File Edit Sketch Tools Help

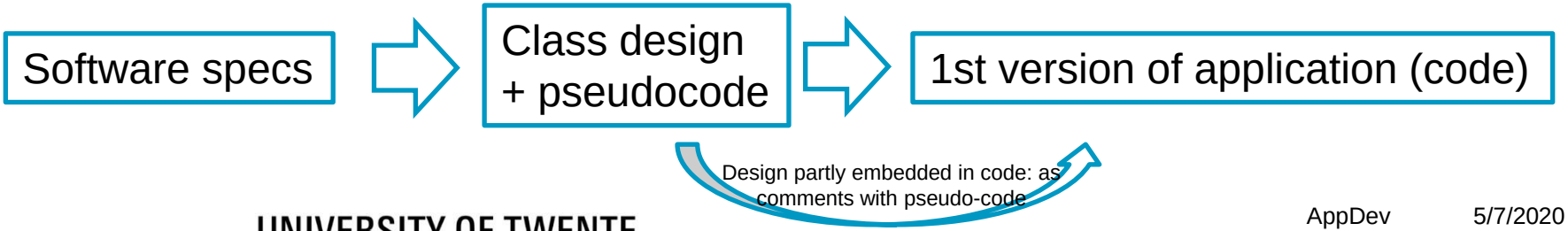
DHT_Unified_Sensor_BLE_chat

```
/**
 * Students must have a short description of the sketch in their own words here.
 * Author: F. van Slooten s1234567
 */

// DHT Temperature & Humidity Sensor
// Unified Sensor Library Example
// Written by Tony DiCola for Adafruit Industries
// Released under an MIT license.

// REQUIRES the following Arduino libraries:
// - DHT Sensor Library: https://github.com/adafruit/DHT-sensor-library
// - Adafruit Unified Sensor Lib: https://github.com/adafruit/Adafruit_Sensor

#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#include <U8g2lib.h>
#include <SoftwareSerial.h>

SoftwareSerial bleserial(0,1); // RX, TX of BLE module

#define DHTPIN 2     // Digital pin connected to the DHT sensor
// Feather HUZZAH ESP8266 note: use pins 3, 4, 5, 12, 13 or 14 --
// Pin 15 can work but DHT must be disconnected during program upload.

// Uncomment the type of sensor in use:
#define DHTTYPE    DHT11     // DHT 11
//#define DHTTYPE    DHT22     // DHT 22 (AM2302)
//#define DHTTYPE    DHT21     // DHT 21 (AM2301)

// See guide for details on sensor wiring and usage:
//   https://learn.adafruit.com/dht/overview
```

2 Arduino Nano, ATmega328P (Old Bootloader) on COM5

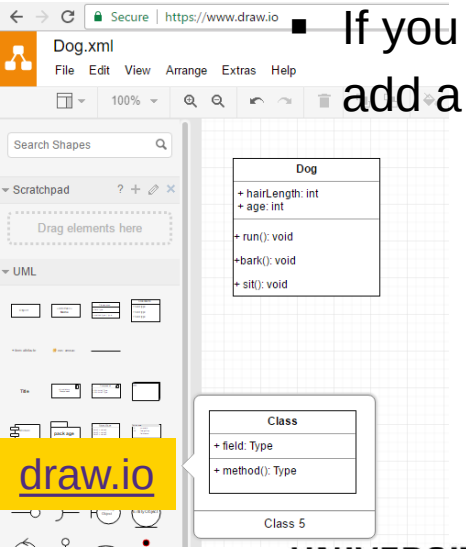
FROM SPECS, TO DESIGN, TO CODE

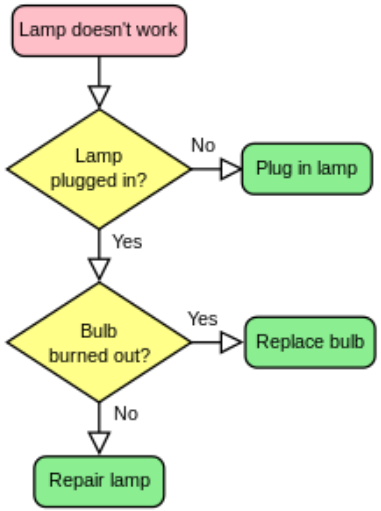
- How do I design an application? Check out previous presentations & assignments
- Results of design-phase:
  - Pseudo code
  - Sketches
  - Diagrams (class-diagram, flowchart, …)

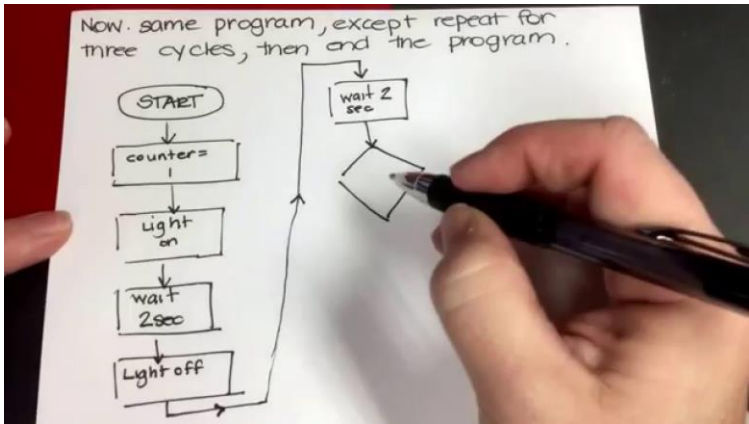


DIAGRAMS

- Can enhance the quality of your design
- If you have multiple classes, create a **class-diagram**
- If you have a complicated flow (of conditions, loops, etc.) add a **Flowchart**







FINAL PRODUCT? OR PROTOTYPE?

- Project is focused on prototype (but it should match final product as close as possible)
- @Application Development, design and code is restricted to prototype only
- This means, for the prototype you can:
  - Eliminate/simplify things you can not build
  - Simulate/demonstrate if necessary

  } only if your tutor agrees on this
- You may show design sketches of final product, as clarification

convince & attract

A prototype demonstrates design to client/ potential customers. And/or you use it for usability tests (goal: prove it can work).

DESIGN A CLASS 'PRODUCT'

IN C++ (ARDUINO)

- Read assignment: we are going to make a *simplified* version of a vending machine…
- … which sells 'Products', which only needs (displays) the name, and does calculations with the price

```
class Product {
private:
   String name;
   int price; // in cents
public:
   // constructor (fills in name and price)
   Product(String n, int p);
   // methods (getters)
   String getName();
   int getPrice();
}
```

Step 1b: Identify methods …

Step 1: *analyze object* in real world…:
It is: a product
Map to properties: color, contents, type, name, dimensions, price…

DESIGN A CLASS 'PRODUCT'

Add an include line for "Product.h"

```
#include "Product.h"

// constructor:
Product::Product(String n, int p) {
  // assign values to class-variables name and price
  name = n;
  price = p;
}

// methods:
String Product::getName() {
  // return the name
  return name;
}

int Product::getPrice() {
  // return the price
  return price;
}
```

Step 2: add pseudo code which describes what methods should do
Step 3: add code to methods (change pseudo code into real code)

ARDUINO: BASED ON C, C++

DIFFERENCES WITH JAVA

- You can use functions without a class:

You may consider a function is ‘a method without a class’

Standard functions `setup()` and `loop()`

```
void setup(){
  Serial.begin(9600);
}

void loop() {
  int sensorValue = readSensor();
  delay(500);
}

int readSensor(){
  int result = analogRead(A0);
  return result;
}
```

- Group code, use (call) from multiple locations

Added function `readSensor()`

arduino.cc/en/Reference/FunctionDeclaration

A CLASS IN C++

DIFFERENCES WITH JAVA

C++ is used in Arduino IDE

Java:

Dog.java

```
class Dog {
   // properties:
   int hairLength;
   int age;

   // methods:
   public void run() {
   }
   public void bark() {
   }
   public void sit() {
   }
}
```

Arduino/C++:

Dog.h

```
class Dog {
   // properties:
   int hairLength;
   int age;

   // methods declaration:
public:
   void run();
   void bark();
   void sit();
};
```

Declaration

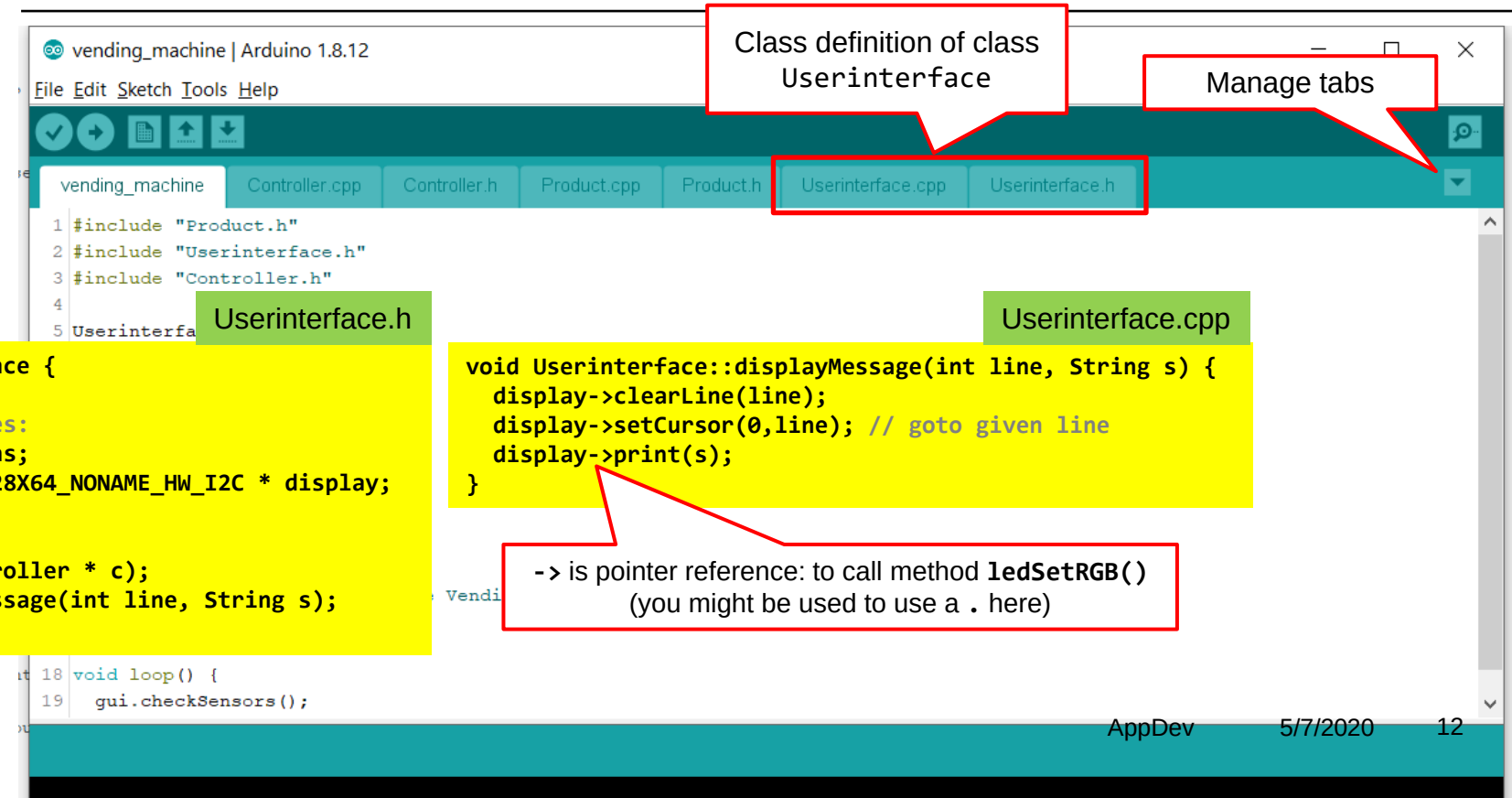
Dog.cpp

```
// methods definition:
void Dog::run() {
}

void Dog::bark() {
}

void Dog::sit() {
}
```

Method definition

C++

COMPARED TO JAVA



CLASS A USES CLASS B

This does not work: "class name does not name a type"
(chicken-and-egg situation)

```
#include <B.h>

class A {
 public:
    A(int id);
 private:
    int id;
    B * objectB;
};
```

```
#include <A.h>

class B {
 public:
    B(int id);
 private:
    int id;
    A * objectA;
};
```

'someFunction' was not declared in this scope
+solution

Solution: use *forward declaration:*

```
class B;

class A {
 public:
    A(int id);
 private:
    int id;
    B * objectB;
};
```

```
class A;

class B {
 public:
    B(int id);
 private:
    int id;
    A * objectA;
};
```

Google the error! > stackoverflow.com/questions/3608305/class-name-does-not-name-a-type-in-c

LISTS

ALSO CALLED: ARRAYS

- List of primitive types
- E.g. `byte, int, double`
- Java:
  - Always initialize with new
  - Lists of objects: use class **`ArrayList`**

Java:

```
// declare list of 4 integers:
int[] list = new int[4];

// assign values:
for (int i = 0; i < list.length; i++) list[i] = i;

int sum = 0;
for (int e : list) sum += e;
System.out.println(sum);
```

Arduino/C++:

```
// declare list of 5 led-pins:
byte leds[] = {2, 3, 4, 5, 13 };

// turn on the second led in the list:
digitalWrite(leds[1], HIGH);

// turn leds on one by one:
for (int i = 0; i < sizeof(leds); i++) {
   digitalWrite(leds[i], HIGH);
   delay(1000); // wait 1 sec.
}
```

Used in test-sketch for assignment 5a.

tutorialspoint: java arrays

ARRAYLIST

ONLY IN JAVA!

Practice: next assignment

- Keeps list of objects
- Assignment 5b: used in Controller
- Methods:

  add(Object)

  get(int)

  size()

```
// declare list which contains balls:
ArrayList<Ball> balls;

// create new ball b:
Ball b = new Ball();

// add ball b to list:
balls.add(b);

// get ball number 5 from the list:
Ball b = balls.get(5);

// how many balls in the list?:
System.out.println("Number of balls: "+balls.size() );
```

ARRAYLIST

JAVA VERSION

Practice: this assignment, given code of Controller has list of products

```
ArrayList<Product> products; // list of products

products = new ArrayList<Product>(); // new empty list

// add a product:
products.add(new Product("Cola", 100));

// method:
private boolean checkPayment() {
   // Get product from list:
   Product p = products.get(chosenItem);

   // If balance is higher or equal to price of chosen product, return true
   if (balance >= p.getPrice()) {
      return true;
   }
   gui.displayMessage("Balance insufficient.");
   return false;
}
```

In assignment: check methods of class Product to get name and price of a Product

ARRAY OF OBJECTS

C++ VERSION

Practice: this assignment, given code of Controller has list of products

```
Product * products[NUM_PRODUCTS]; // list of products, new empty list

// add a product:
products[0] = new Product("Cola", 100);

// method:
bool Controller::checkPayment() {
    // Get product of list:
    Product * p = products[chosenItem];

    // If balance is higher or equal to price of chosen item, return true
    if (balance >= p->getPrice()) {
      return true;
    }
    gui->displayMessage("Balance insufficient.");
    return false;
}
```

In assignment: check methods of class Product to get name and price of a Product

LOOP TROUGH A LIST

JAVA VERSION

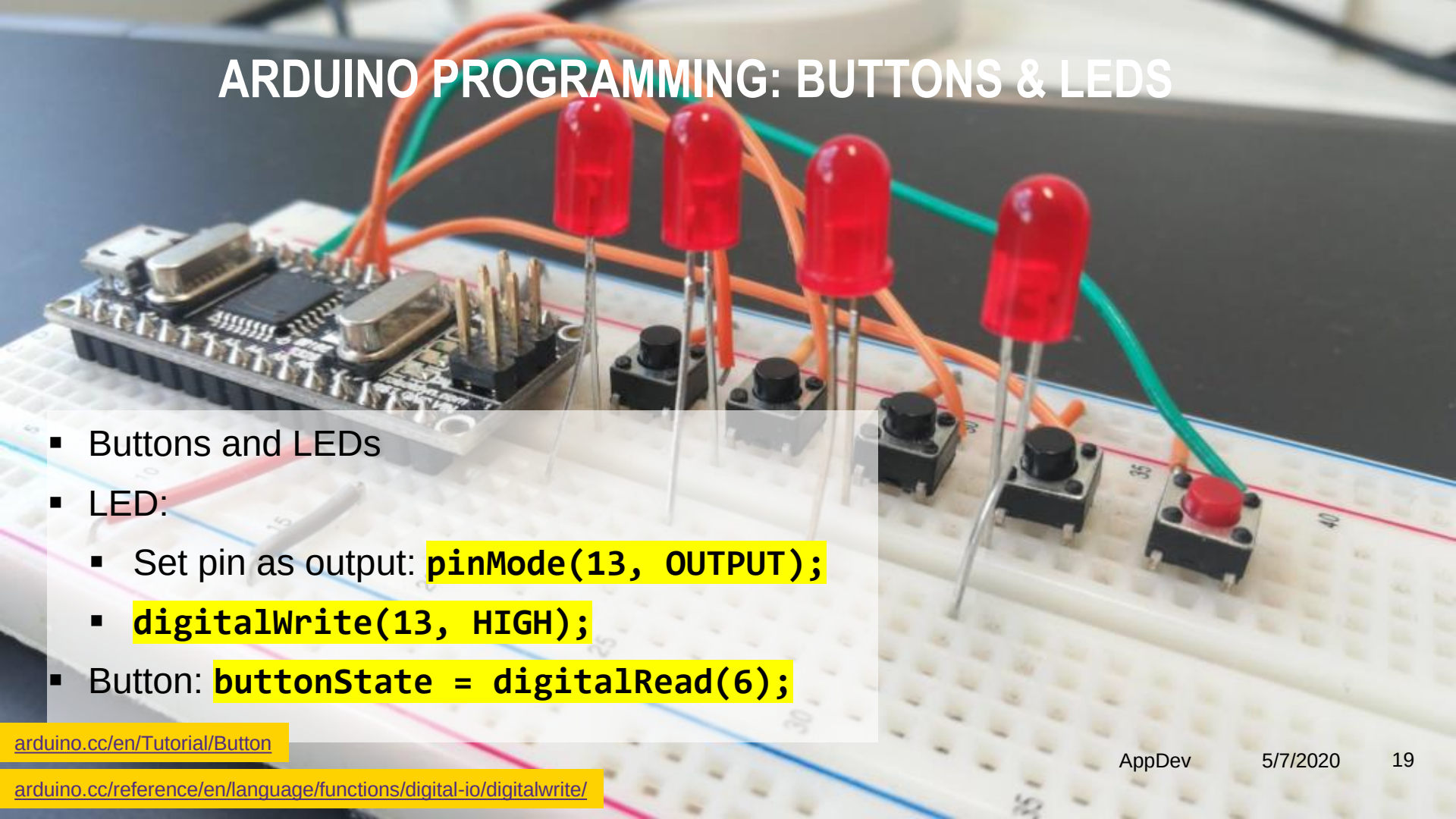
- for-each loop:

```
public void paintComponent(Graphics g) {
  super.paintComponent(g);
  // Draw all balls by calling the paintComponent() method for each ball:
  for (Ball b : balls) { // for each ball in the list..
      b.paintComponent(g); // draw the ball
  }
}
```

- ‘normal’ for loop:

```
for (int i = 0; i<balls.size(); i++) {
   balls.get(i).paintComponent(g);
}
```

ARDUINO PROGRAMMING: BUTTONS & LEDS



- Buttons and LEDs
- LED:
  - Set pin as output: `pinMode(13, OUTPUT);`
  - `digitalWrite(13, HIGH);`
- Button: `buttonState = digitalRead(6);`

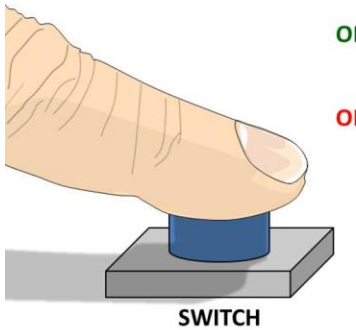
arduino.cc/en/Tutorial/Button

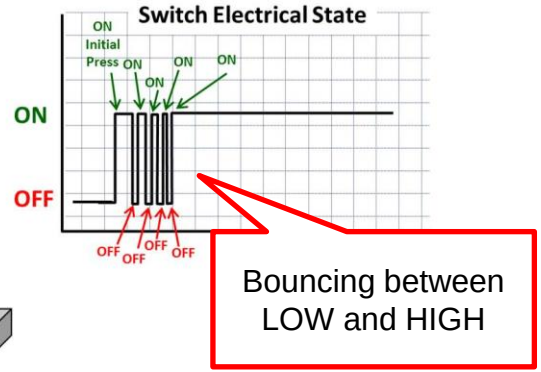
arduino.cc/reference/en/language/functions/digital-io/digitalwrite/

PRESSING A BUTTON…

Example: multi_button_check_analog_input_oled_display.ino

- Interaction with button takes time:
- Switching from LOW to HIGH has a short period in which state is undefined
- We need to ‘de-bounce’
- In assignment, we use Bounce2-library for this





A LOT OF BUTTONS…

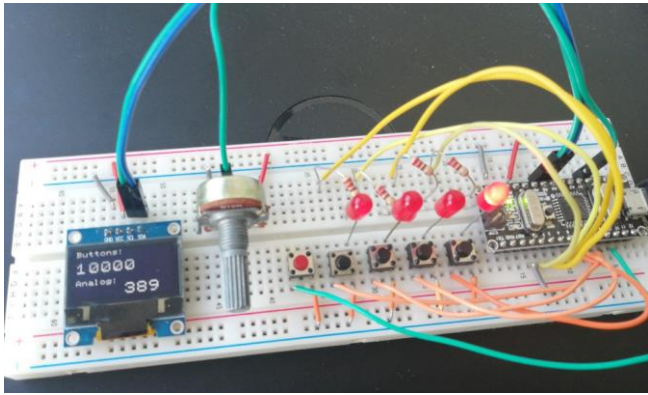
Use Bounce2 library

List of pin numbers

List of buttons

Setup the buttons in a for-loop

```
#include <Bounce2.h>

byte button_pins[] = {6, 7, 8, 9, 10 }; // button pins

#define NUMBUTTONS sizeof(button_pins)

Bounce * buttons = new Bounce[NUMBUTTONS];

void setup() {
  // Make input & enable pull-up resistors on switch pins
  for (int i=0; i<NUMBUTTONS; i++) {
    // setup the bounce instance:
    buttons[i].attach( button_pins[i], INPUT_PULLUP);
    buttons[i].interval(25); // interval in ms
  }
}
```



We use internal pull-up* resistors of Arduino, this saves us having to mount 5 additional resistors

* 'pull-up' means a resistor pulls-up the signal (so it is HIGH).
This also means we have to check the value to fall (becomes LOW) when the button is pressed.

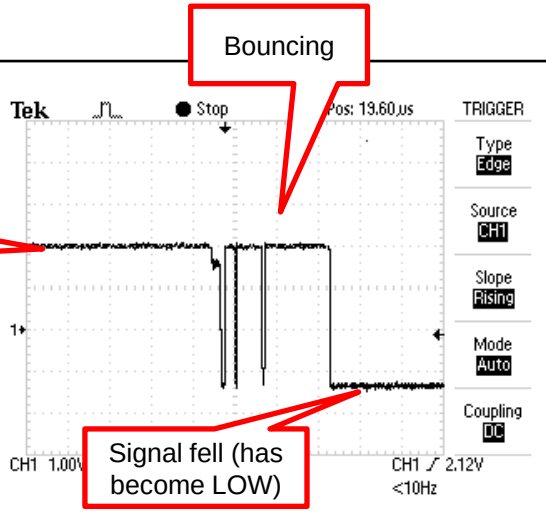
CHECKING A LOT OF BUTTONS…

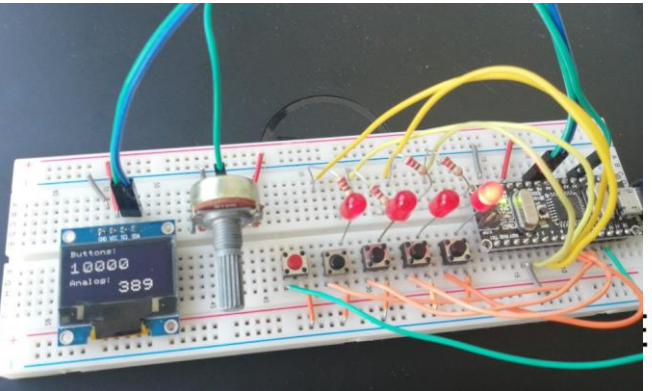
```
void loop() {
  // check all buttons if they are pressed:
  for (int i = 0; i<NUMBUTTONS; i++) {
    // Update the Bounce instance:
    buttons[i].update();
    // If it fell*, do something:
    if ( buttons[i].fell() ) {
      Serial.print(i);
      Serial.println(" press");
    }
  }
}
```

For-each button…

Check if the signal from the button fell*

Signal is HIGH (due to pull-up)

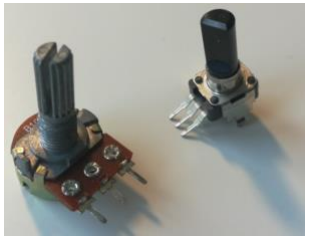


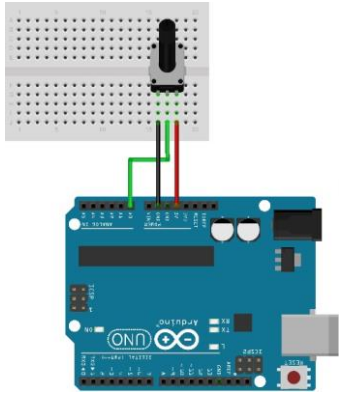


* Because pull-up resistors were used, we must check the value to fall (becomes LOW) when the button is pressed.

READING ANALOG VALUE

- Potmeter: variable resistor
- `sensorValue = analogRead(A0);`





```
void loop() {
  // read analog input:
  sensorValue = analogRead(sensorPin);
  // check if value changed from last read:
  if (abs(lastSensorValue-sensorValue)>2 ) {
    lastSensorValue = sensorValue;
    Serial.print("Analog val: ");
    Serial.println(sensorValue);
  }
}
```

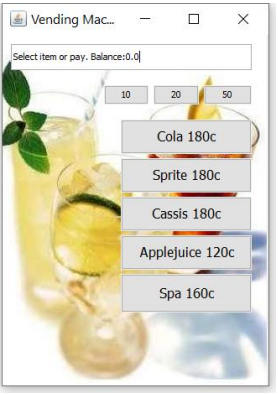
wikipedia.org/wiki/Potentiometer

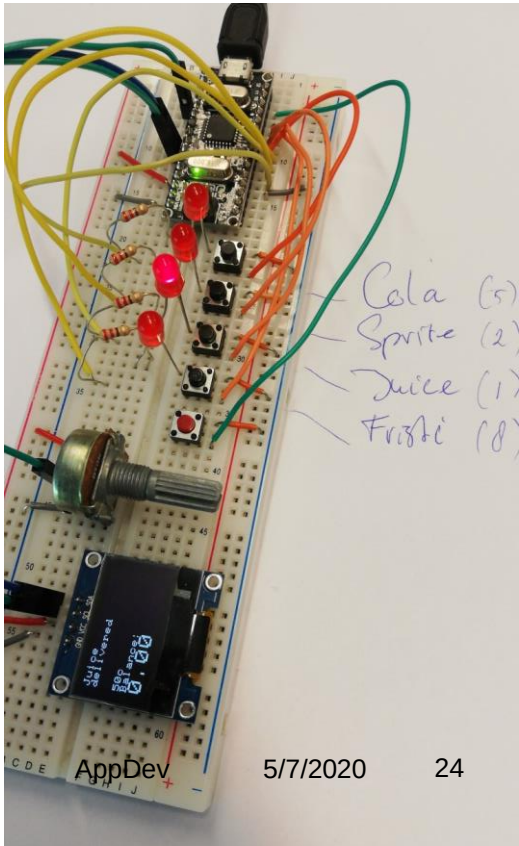
arduino.cc/reference/en/language/functions/analog-io/analogread/

ASSIGNMENT #4

This afternoon: teacher available for help with project via chat, or make appointment for Skype session

- “Build prototype of soda machine”
- a-version:
  - Build electronic circuit with Arduino
- b-version:
  - Create userinterface with Eclipse





Use website chat/forum if you need help!

Slides, assignments etc @ vanslooten.com/appdev