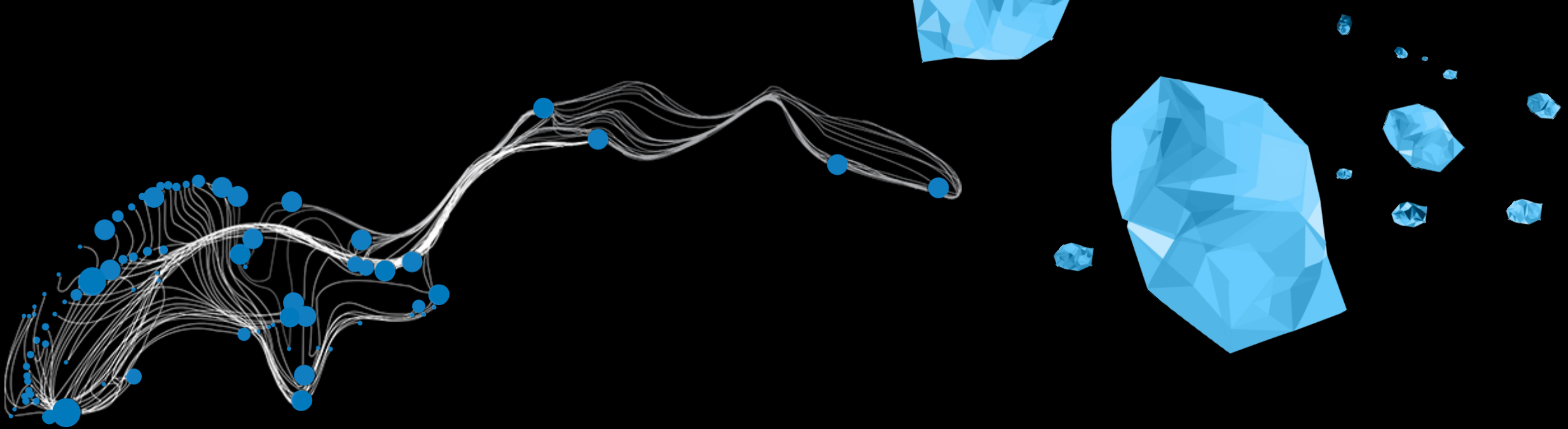
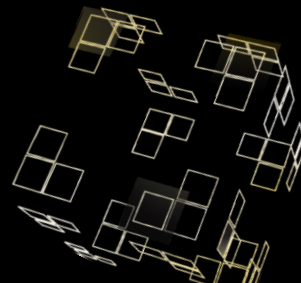
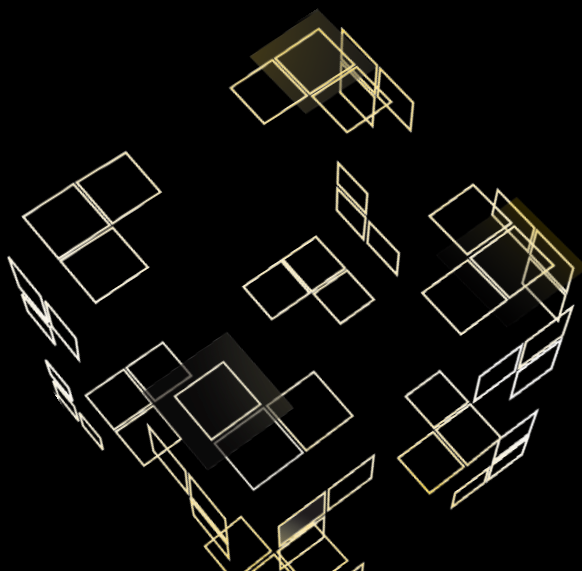




Workshop “prototyping”

By Fjodor van Slooten



**UNIVERSITY
OF TWENTE.**

Workshop prototyping

- Recap App Development M4
- Physical prototypes
 - Building connected sensors using an ESP module (Wifi)
 - Control electronics from an Android App via Bluetooth
- App Mockups ('Virtual' prototypes)
- Intro to Mobile App Development

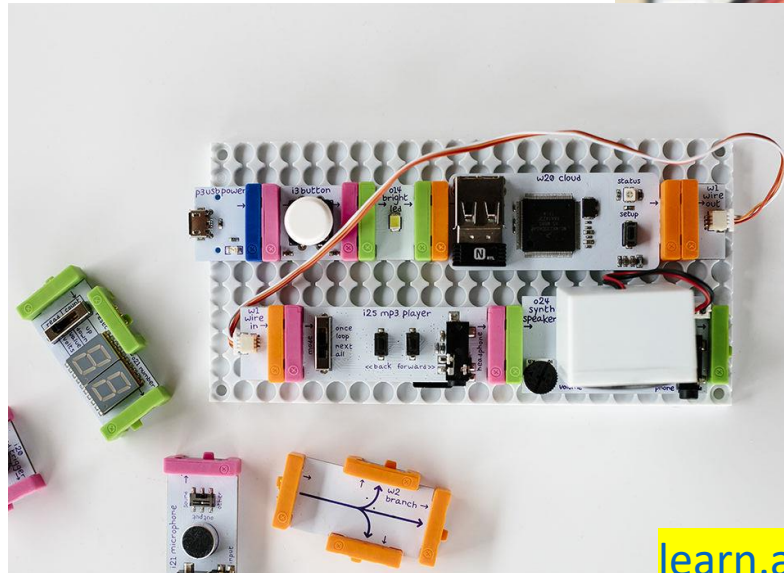
Including 'hands-on' tutorials
you can work on today

Contact: f.vanslooten@utwente.nl, W241

This presentation & tutorials available at: vanslooten.com/workshop

RECAP

- Physical prototypes
 - Cardboard, wood, paper
 - Electronics: Arduino
 - Lego Mindstorms
 - LittleBits



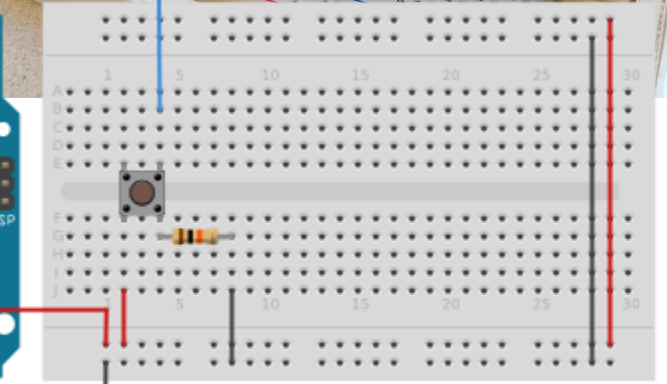
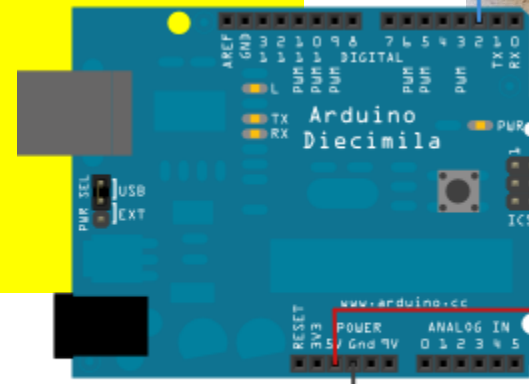
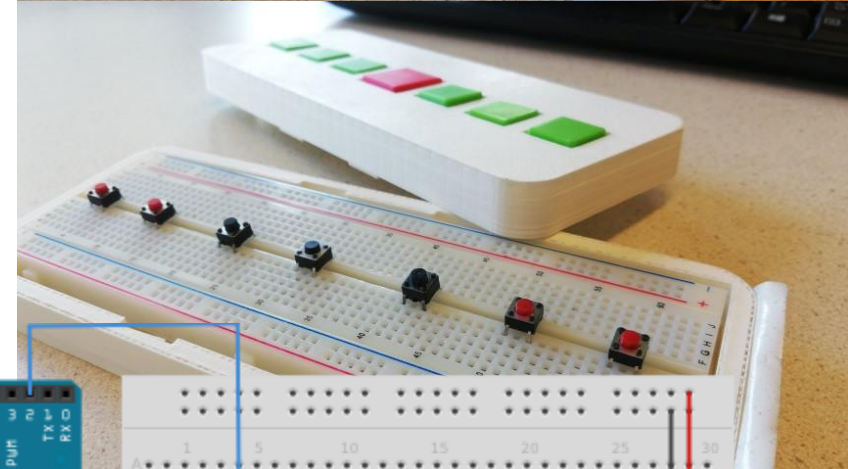
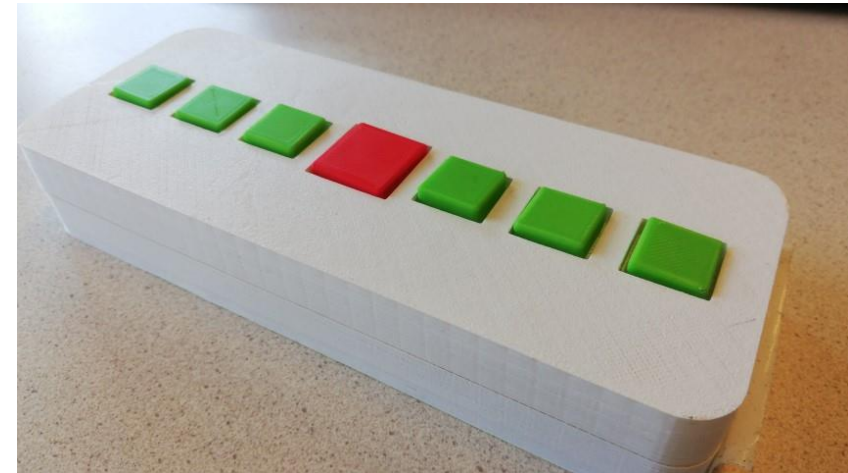
learn.adafruit.com/adafruit-arduino-lesson-2-leds/breadboard-layout

Arduino programming: a button

[simplest kind of sensor]



```
// 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);  
}
```

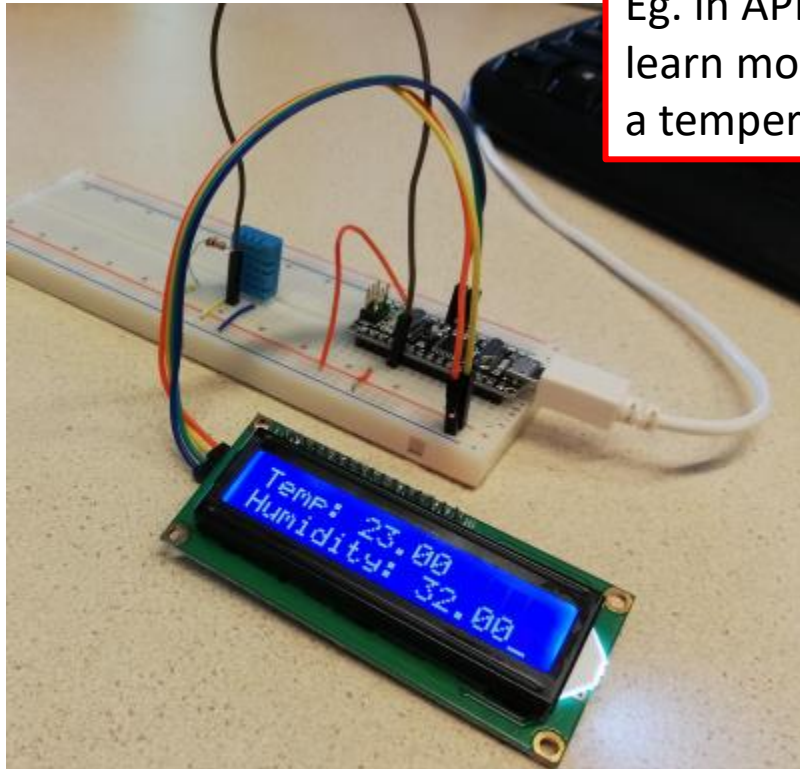


arduino.cc/en/Tutorial/Button

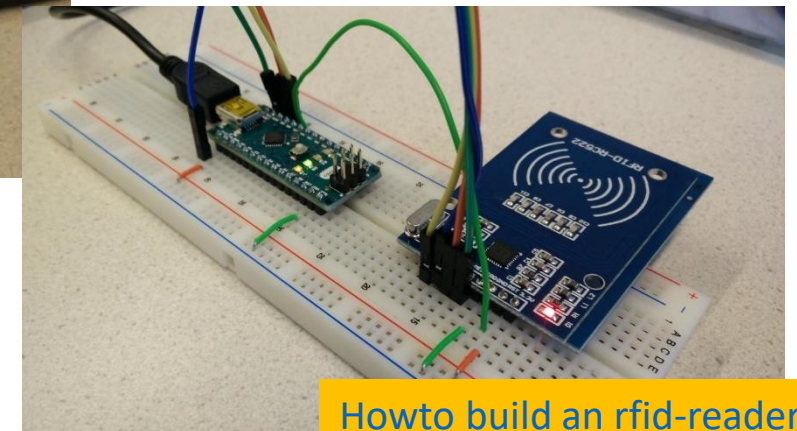
Arduino lessons

Learn more on Arduino programming, with lots of examples at [Application Development \(APPDEV\) course](#)

Eg. in APPDEV Lesson [LegoPracticalSession3](#),
learn more about: use of displays and how to use
a temperature sensor



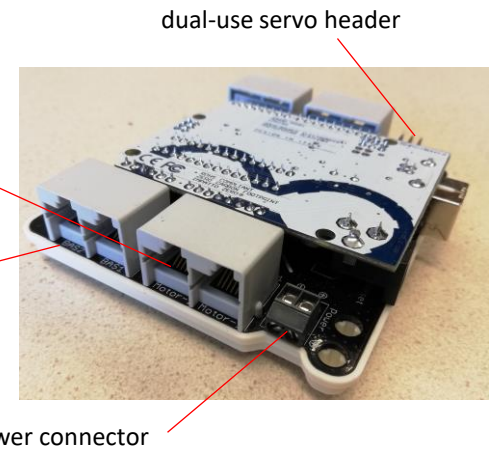
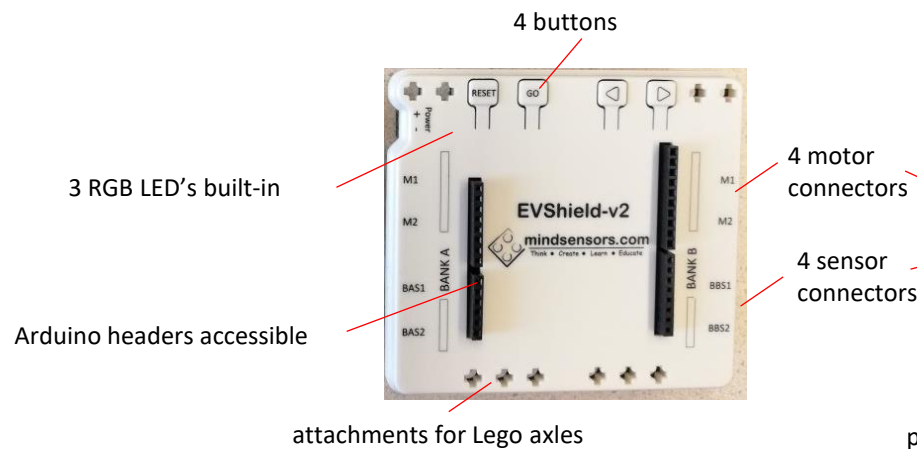
[Touch screen examples:](#)
[keypad](#), [paint](#), [image viewer](#)
and [calculator](#)



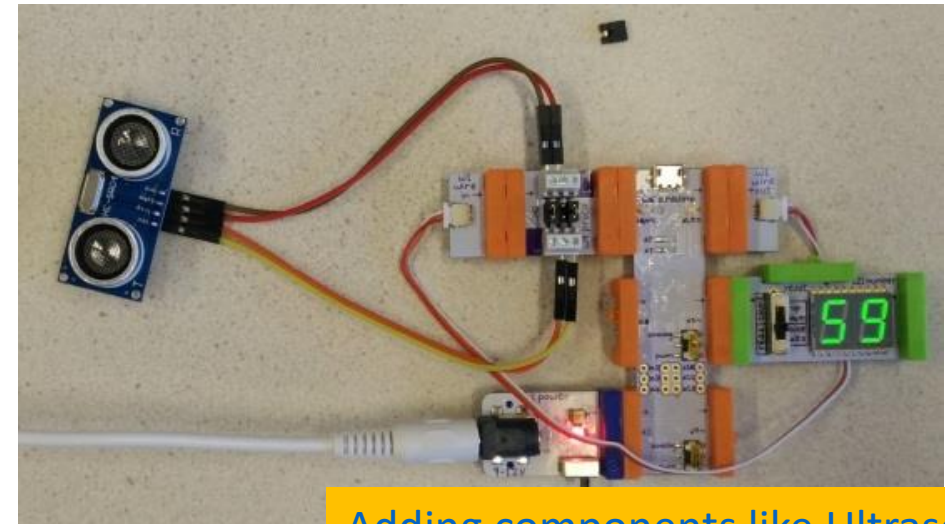
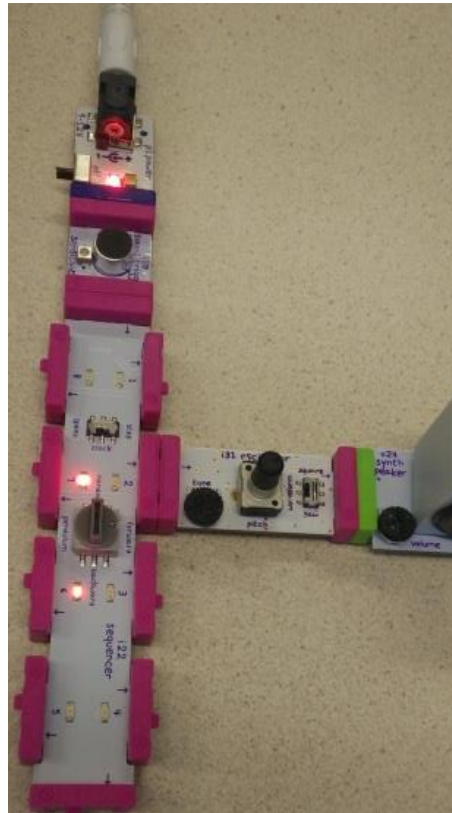
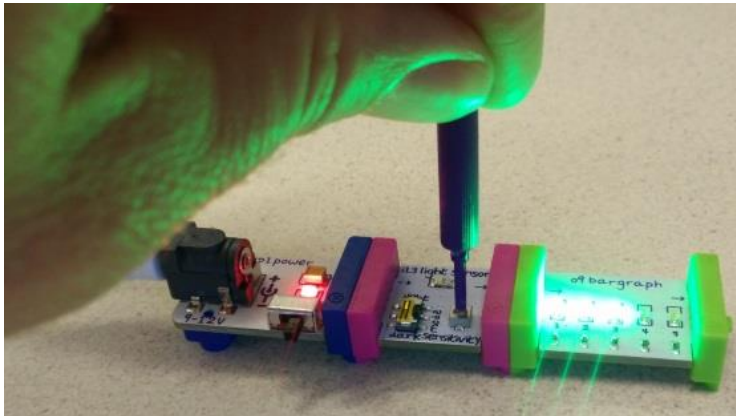
[Howto build an rfid-reader](#)

EVShield: control Lego Mindstorms with Arduino

- Connect (almost) all Lego motor's and sensors



LittleBits Examples



[Adding components like Ultrasonic sensor to LittleBits](#)



[Getting started with Arduino using LittleBits](#)

[LittleBits resources & interaction examples](#)

App mockups

Start creating a prototype (or mock-up) as soon as possible
... and involve the user from **day 1**!



[Axure tutorial](#)

Tools

- [Sketch](#) (Mac only)
- [InVision](#) (Online, free with limitations)
- [Figma](#) (Online, free with few limitations)
- [Axure](#) (license available from teacher)
- [Adobe XD](#) (free)

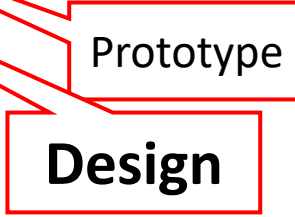
Mock-up = a prototype (of a user interface)
for usability tests

[Learn more about Userinterface Design](#)

The screenshot shows the Adobe Photoshop toolbar at the top. The 'Shape' tool icon (a square) is selected, and its dropdown menu is open. The menu lists the following tools with their respective keyboard shortcuts:

- ✓ Rectangle R
- Line L
- Arrow Shift+L
- Ellipse O
- Polygon
- Star
- Place Image Ctrl+Shift+K

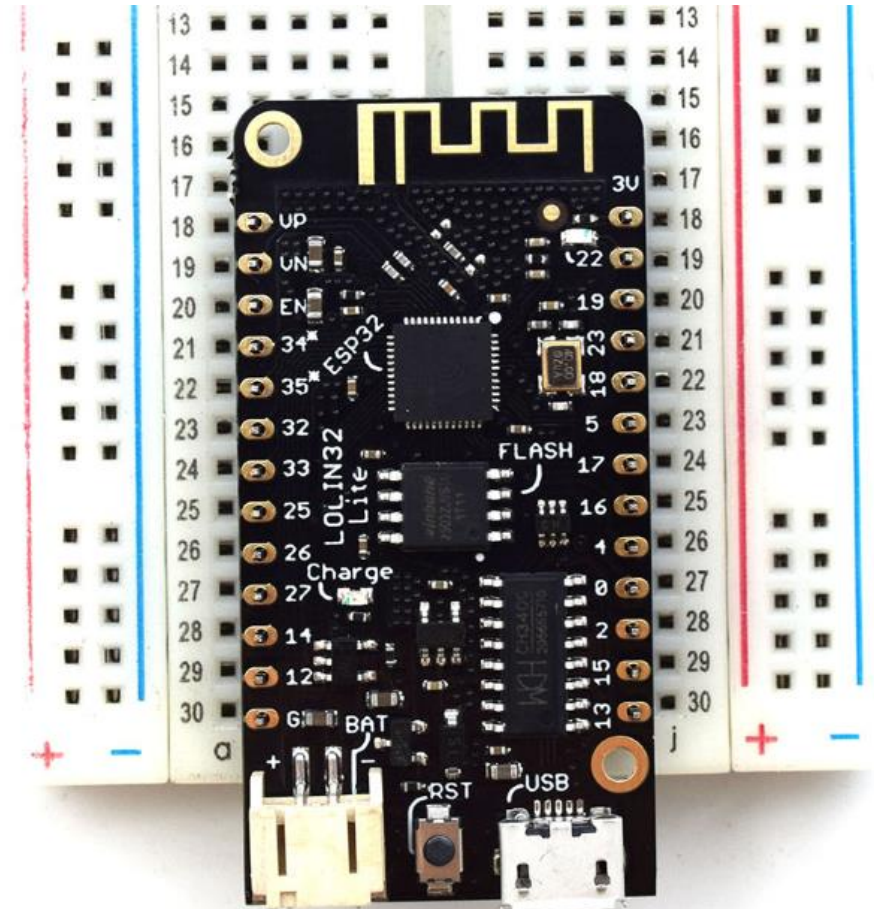
The 'Layers' panel on the left shows a single layer named 'Page 1'. The 'Pages' panel shows a single page thumbnail. The 'Properties' panel on the right is empty. The 'Code' panel at the bottom left shows the HTML code for the page.



[Figma tutorial: create a first app mockup](#)

PHYSICAL PROTOTYPES

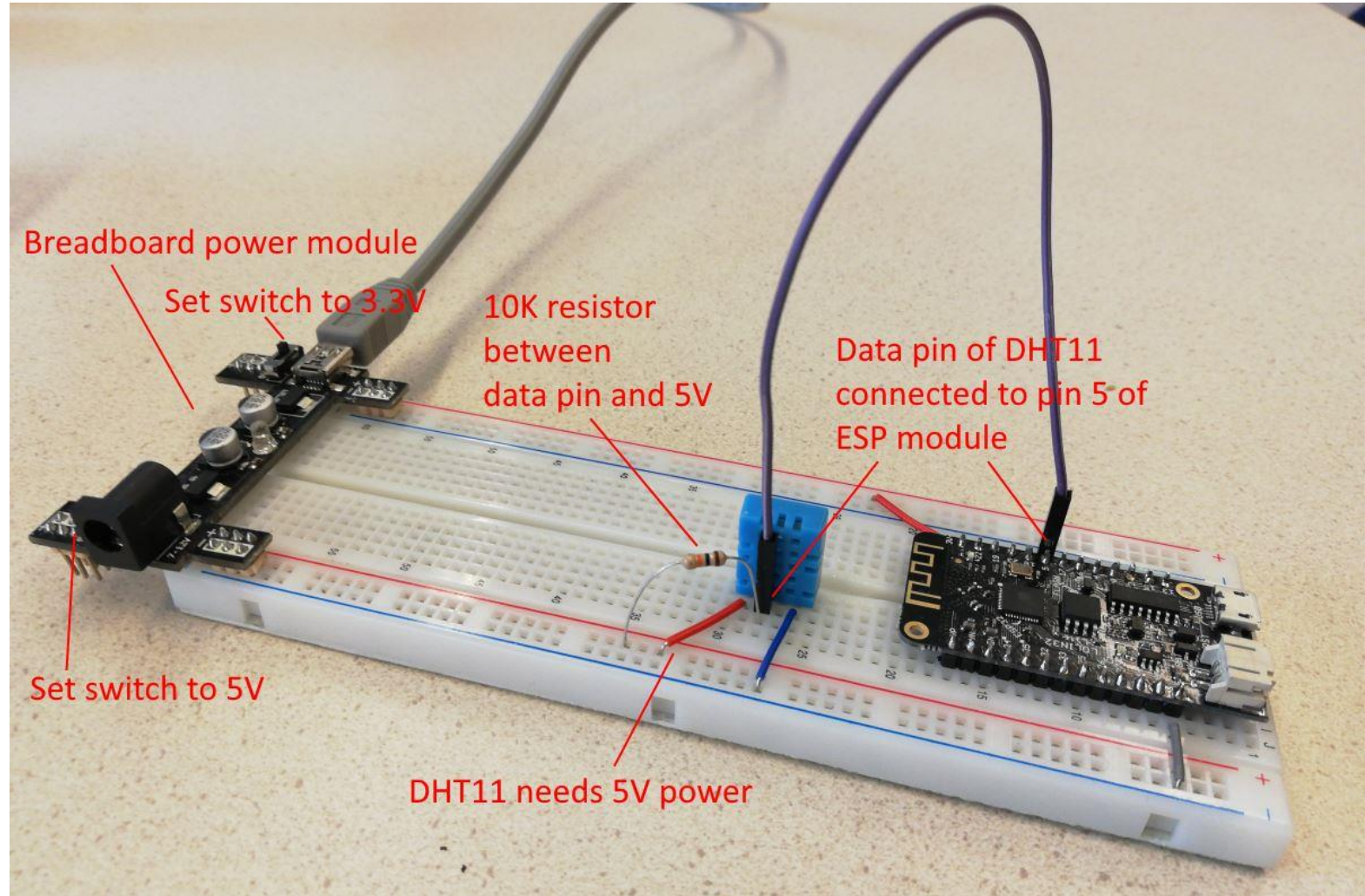
- Building connected sensors using an ESP module
- What is an ESP module?
 - A small microcontroller (~ computer)
 - Can connect & control electronic circuits
 - Has Wifi (and sometimes Bluetooth)
- Combination of electronics & wireless communication



Connect DHT11 temperature sensor to ESP32 module

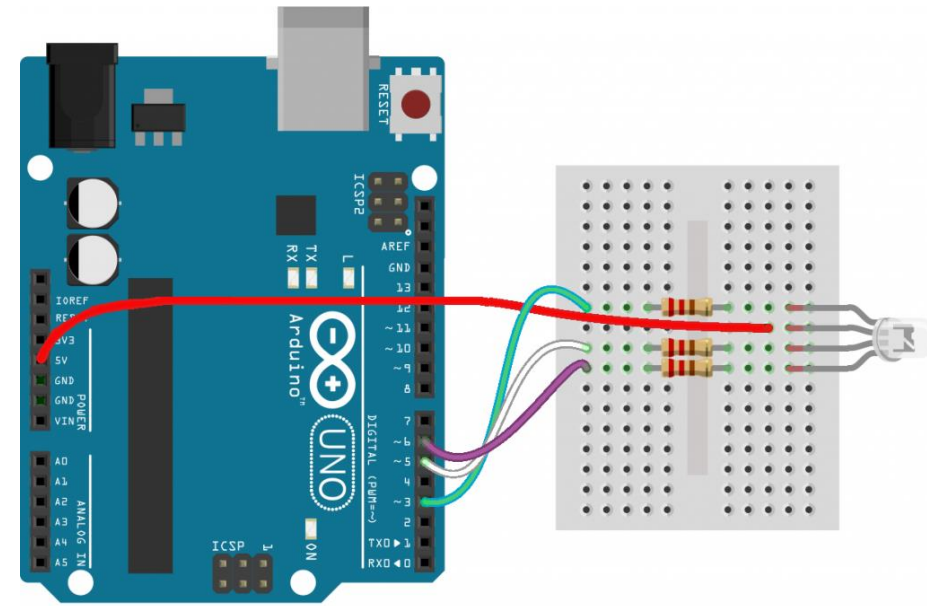
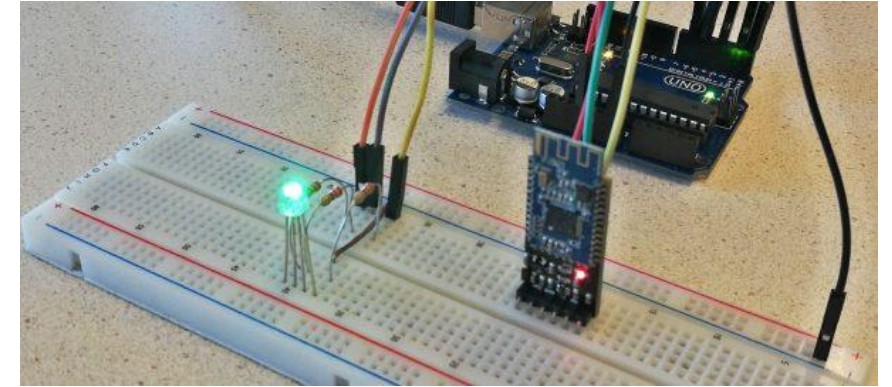
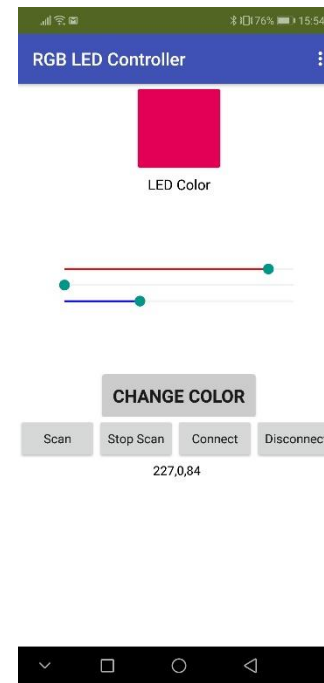
- Build this: [Tutorial: Create a connected sensor](#)
- It can:
 - Read temperature & humidity
 - Connect to WiFi
 - Store values online

Alternative: use [Blynk](#) cloud service



Control an RGB LED from an Android App via Bluetooth

- Build this: “Control an RGB LED from an Android App via Bluetooth”
- It can:
 - Change color of LED from an App
 - Connect via Bluetooth to a phone



MOBILE APP DEVELOPMENT

- Types of Apps
 - Web (Webview)
 - Hybrid
 - Native
- Development environment
 - Online
 - On device (phone or tablet)
 - On desktop/laptop

Web Apps are coded in HTML/CSS/JavaScript.

They are served through the internet and run through a browser.

- +/- Access Native APIs
- Distribute through App Stores
- + Run on multiple platforms

Hybrid Apps are coded in HTML/CSS/JavaScript.

They are run through an invisible browser (webview) that is packaged into a native application.

- + Access Native APIs
- + Distribute through App Stores
- + Run on multiple platforms

Native Apps are coded in the native language of the device (e.g. Objective-C for IOS, Java for Android). They are run directly on the device.

- + Access Native APIs
- + Distribute through App Stores
- Run on multiple platforms

App generators

- [App Inventor](#) (online service)
- [Sketchware](#) (create Apps on your phone)
- [Tasker+Tasker App Factory](#) (Paid App)



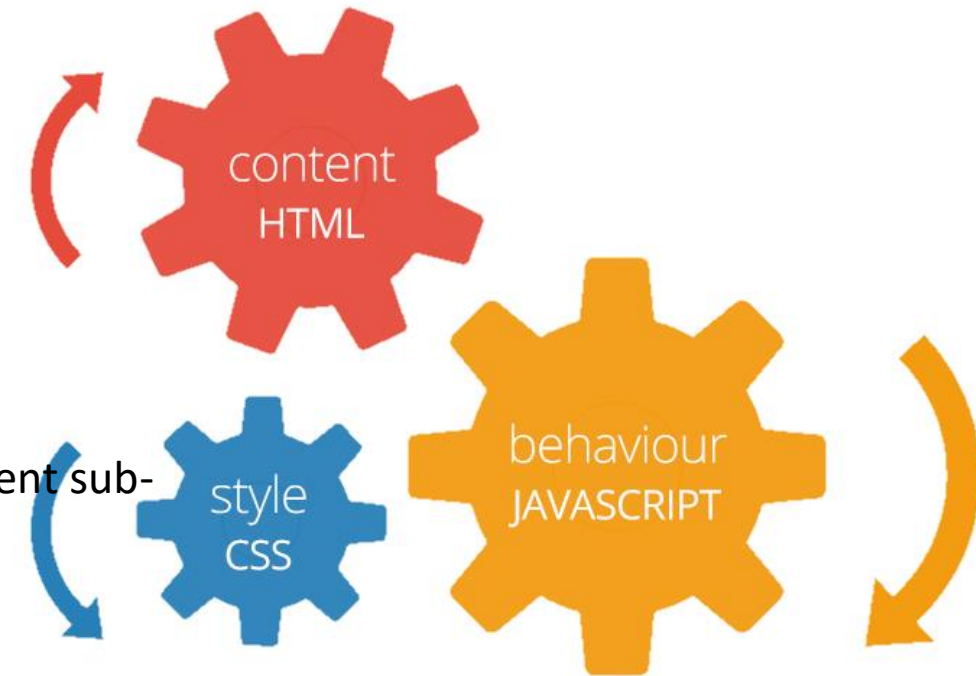
Sketchware and App Inventor use **Block Language** programming

Graphical User Interface Builder



Hybrid Apps

- Lots of competition
- HTML/CSS/JavaScript based
- Often immature (version hell, poor documentation, different sub-platforms, mixed-up online info)
- Requires: intermediate programming knowledge, and web tech (HTML/CSS/JavaScript)
- Tools: Code editor/IDE e.g.: [Atom](#), [Visual Studio Code](#), [Brackets](#)
- Learn basics: course [Web Tech](#), [App Dev](#), [w3schools](#)



Hybrid Apps Frameworks

Based on Web Technology

- [Ionic](#)
- [NativeScript](#) (delivers Native App)
- [PhoneGap](#) (Adobe)
- [React Native](#) (delivers Native App)
- [Xamarin](#) (Microsoft, C#) (delivers Native App)
- [jQuery Mobile](#)

Ionic 	NativeScript 
Quick build & test	Build/test takes longer (wait each time you change)
Preview in browser or on Phone	Preview on Phone or Emulator
App: renders in Webview	Converts to native (small performance gain)
Create UI based on HTML/CSS	Create UI with own elements

Native apps

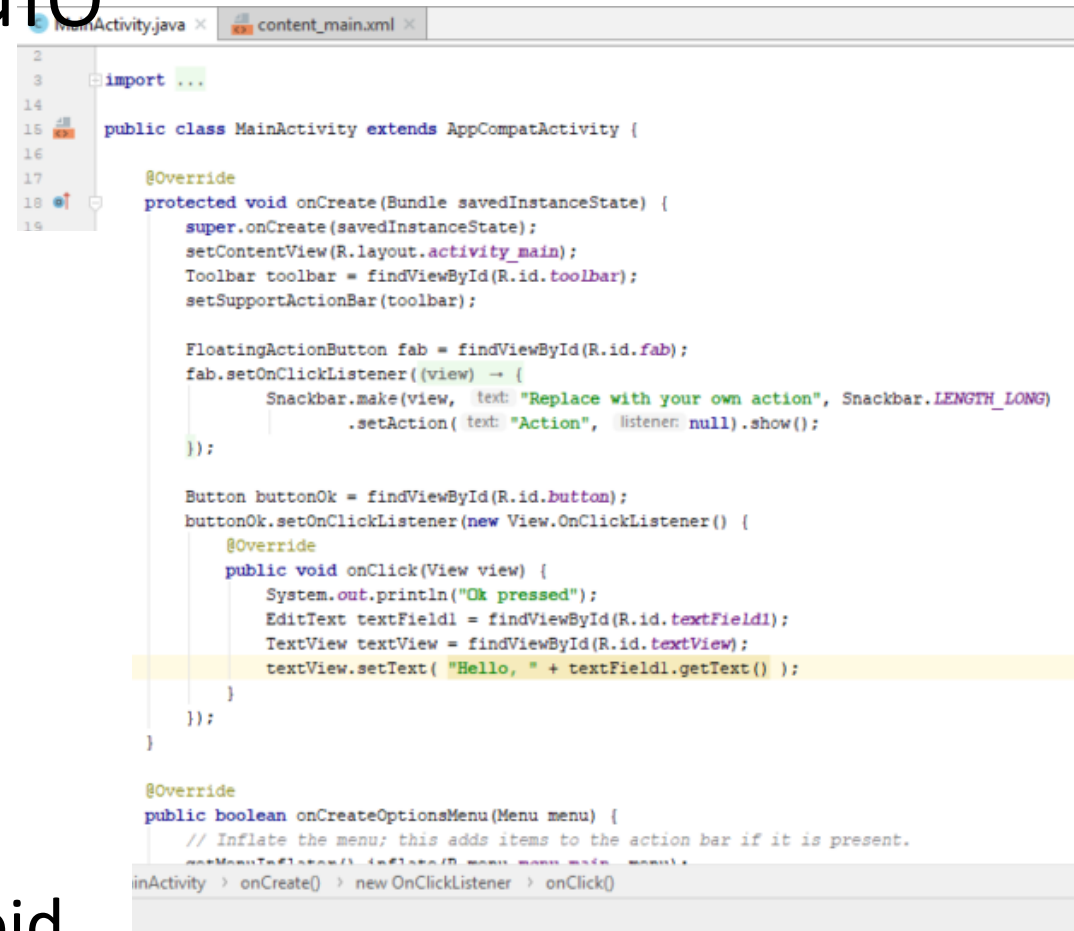
developer.android.com
developer.apple.com

- Platform: IOS (Apple), Android (Google)
- Android:
 - [Andoid Studio](#) (Java)
- IOS
 - [Xcode](#) (Objective-C)
 - [Swift](#)
- Requires: programming knowledge, and preferably web tech
- Learn IOS: [Everyone can code](#), [App development with Swift](#)



Android Studio (Java)

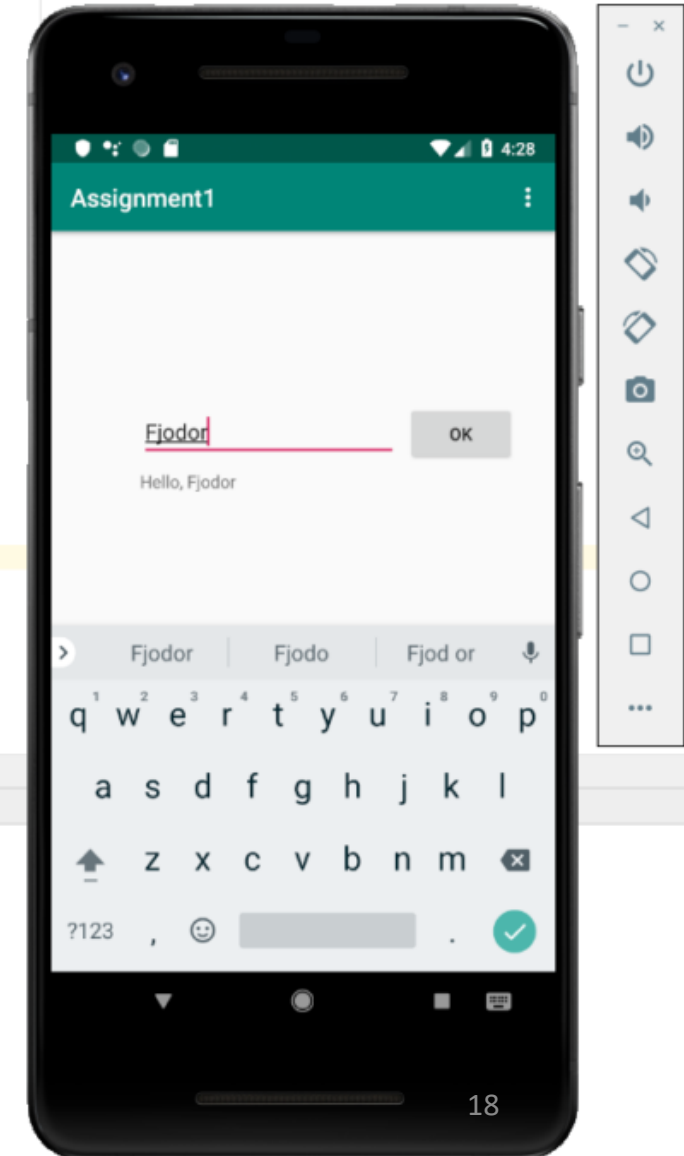
- Tutorials:
 - [Android Developer fundamentals](#)
 - developer.android.com
 - [more](#)
- Requires: Java programming experience
- Learn basics of Android Studio:
 - [Getting started](#)
 - [Create a first interactive userinterface](#)



```

1  MainActivity.java x content_main.xml x
2
3  import ...
14
15  public class MainActivity extends AppCompatActivity {
16
17      @Override
18      protected void onCreate(Bundle savedInstanceState) {
19          super.onCreate(savedInstanceState);
20          setContentView(R.layout.activity_main);
21          Toolbar toolbar = findViewById(R.id.toolbar);
22          setSupportActionBar(toolbar);
23
24          FloatingActionButton fab = findViewById(R.id.fab);
25          fab.setOnClickListener((view) -> {
26              Snackbar.make(view, text: "Replace with your own action", Snackbar.LENGTH_LONG)
27                  .setAction(text: "Action", listener: null).show();
28          });
29
30          Button buttonOk = findViewById(R.id.button);
31          buttonOk.setOnClickListener(new View.OnClickListener() {
32              @Override
33              public void onClick(View view) {
34                  System.out.println("Ok pressed");
35                  EditText textField1 = findViewById(R.id.textField1);
36                  TextView textView = findViewById(R.id.textView);
37                  textView.setText( "Hello, " + textField1.getText() );
38              }
39          });
40      }
41
42      @Override
43      public boolean onCreateOptionsMenu(Menu menu) {
44          // Inflate the menu; this adds items to the action bar if it is present.
45          getMenuInflater().inflate(R.menu.menu_main, menu);
46      }
47  }
48
49  MainActivity > onCreate() > new OnClickListener > onClick()

```

Publish in App store (deploy)

- Requires:
 - IOS: register (\$99) with developer.apple.com/programs ([more info](#))
 - Android: register (for free) with play.google.com/apps/publish
- Alternative: **test without publishing**
 - Android: distribute generated .apk to testers
 - IOS: have testers get a build [via TestFlight](#): register participants on your team



Game engines

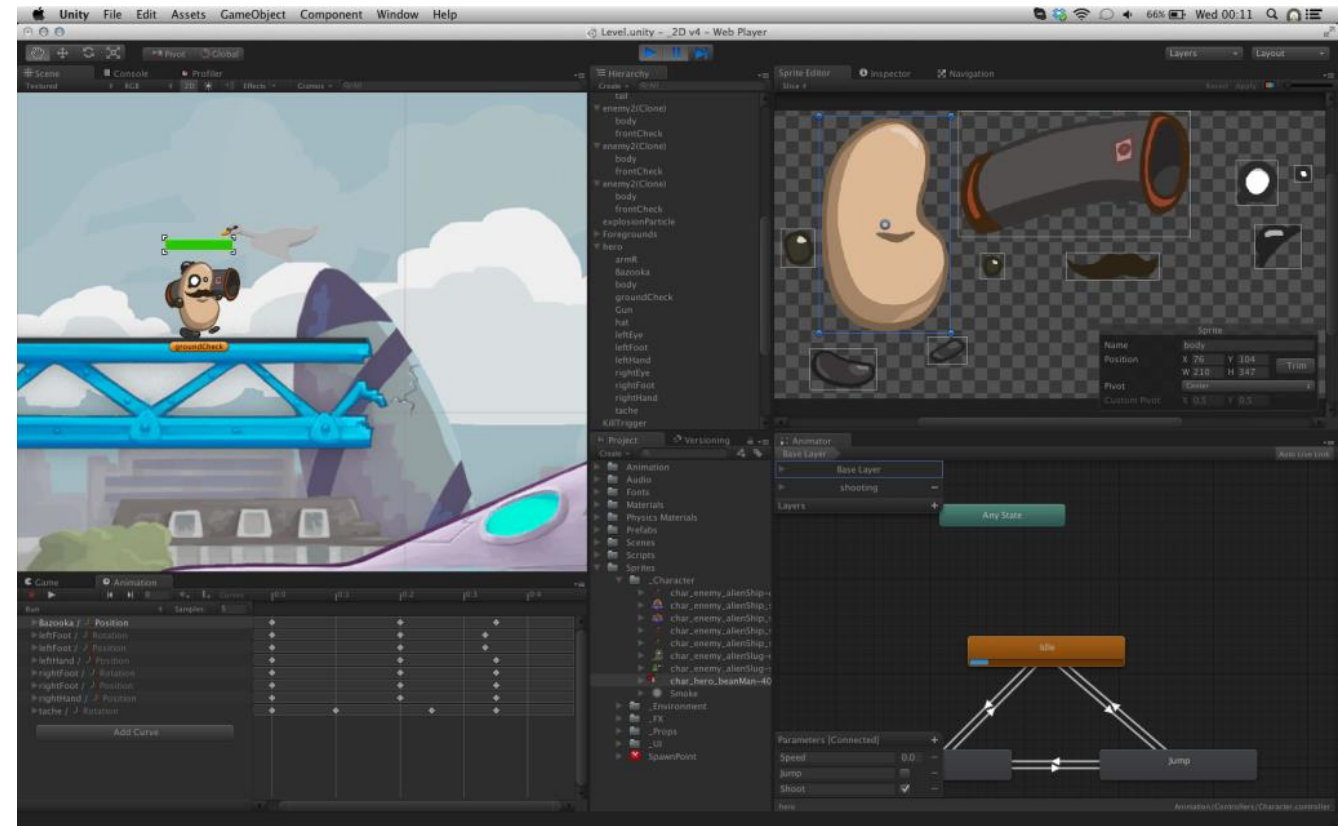
- Combine with Native App:
 - [Unity](#): [Mobile solutions](#)
 - [Unreal](#): [Mobile Development](#)
- Combine with Hybrid framework:
 - [Phaser.io](#):
 - [Deploy with Cocoon.io](#)
 - Combine with: [Ionic](#), [PhoneGap](#)

Learn more about game development:

emanueleferonato.com

html5gamedevs.com

gamedev.net



Intro to App Inventor

Designer **Blocks** Program

Palette

User Interface

- Button
- CheckBox
- DatePicker
- Image
- Label
- ListPicker
- ListView
- Notifier
- PasswordTextBox
- Slider
- Spinner
- TextBox
- TimePicker
- WebView

Layout

Media

Drawing and Animation

Maps

Viewer

Screen1

Image1

ButtonHelloWorld

Height: 300 pixels...

Width: 300 pixels...

Picture: globe.png...

RotationAngle: 0.0

ScalePictureToFit: ☒

Visible: ☒

globe.png

Upload File ...

[App Inventor tutorial](#)
[App Inventor 2: Create your own Android Apps](#)

Programming using Blocks

Designer

The screenshot shows the App Inventor Designer interface. At the top, there's a green header bar with the text "HelloWorld" and buttons for "Screen1", "Add Screen ...", and "Remove Screen". On the right side of the header, there are two tabs: "Designer" (highlighted with a red box) and "Blocks".

Below the header, the interface is divided into three main sections:

- Blocks:** A vertical palette on the left containing various block categories: Built-in (Control, Logic, Math, Text, Lists, Colors, Variables, Procedures), Screen1, Image1, ButtonHelloWorld, and Any component.
- Viewer:** The central area showing the code blocks. It contains two event-driven blocks:
 - when Screen1 .Initialize** with a **do** block: **set Image1 . Visible to false**.
 - when ButtonHelloWorld .Click** with a **do** block: **set Image1 . Visible to not Image1 . Visible**.
- Preview:** A mobile device simulation on the right showing the app's visual design. It includes a status bar at the top with the time 9:48, a header "Screen1", a large image of a globe, and a button labeled "Hello World" at the bottom. The device has standard Android navigation icons at the very bottom.

[App Inventor tutorial](#)
[App Inventor 2: Create your own Android Apps](#)

Tutorial: Creating a connected App

The screenshot shows the MIT App Inventor WebViewer interface. The top bar includes 'WebViewer', 'Screen1', 'Add Screen...', 'Remove Screen', 'Designer', and 'Blocks'. The left sidebar contains a 'Palette' with categories: 'User Interface' (Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, WebViewer), 'Layout', 'Media', 'Drawing and Animation', 'Maps', and 'Sensors'. The central 'Viewer' pane displays a mobile app design with a status bar at the top showing signal, Wi-Fi, and battery icons, and the time 9:48. The app has a title bar 'Weather station' and three text labels: 'Temperature:', 'Humidity:', and 'LastUpdate:'. Below these is a globe icon. The main content area features a house illustration with a red roof, a chimney, and a small tree. The bottom of the app has a black navigation bar with three icons. The right sidebar contains 'Components' and 'Properties' panes. The 'Components' pane lists 'Screen1' containing 'TableArrangement1' with sub-components 'LabelTemp', 'LabelHum', 'LabelTempValue', 'LabelHumValue', 'LabelLastUpdate', and 'WebViewerRefreshTime'. It also lists 'Clock1', 'Web1', and 'Web2'. The 'Properties' pane shows the properties for 'LabelTempValue', including 'BackgroundColor', 'FontBold', 'FontItalic', 'FontSize' (16.0), 'FontTypeface' (default), 'HTMLFormat', 'HasMargins' (checked), 'Height' (20 pixels), 'Width' (100 percent), 'Text', 'TextAlignment' (left: 0), 'TextColor' (Default), and 'Visible' (checked). At the bottom of the 'Components' pane is a 'Media' section with 'weerhuisje.jpg' and an 'Upload File...' button.



[Tutorial: Build an App with App Inventor which can display values of a connected sensor](#)

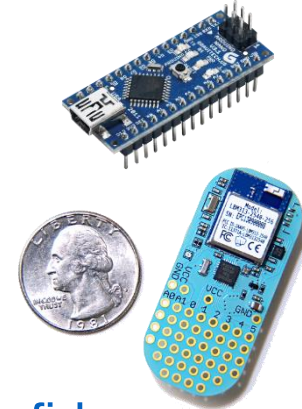
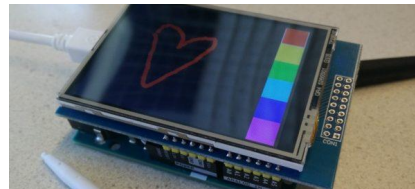
BORROW

MATERIALS

- Arduino** + EL-kit*
- LittleBits kit



- Arduino Leonardo/Nano/Wemos D1, [ESP32](#), [Wifi module](#), [Bluetooth module](#), Blue bean
- [Touchscreen](#), [displays](#)
- Motors, [vibration motor](#)
- [Mp3 module](#), [speakers](#), sd-card
- Sensors: [distance](#), [color](#), [accelerometer/gyro](#), [temp](#), [rfid](#), rotation (in Lego motor)
- [RGB-leds](#)



[Check out the contents of the SmartProducts \(M4\) kit here](#)



Tip: buy cheap components @ **STORES**
[Stores-shop](#), Educafé, Zilverling (Starbucks)

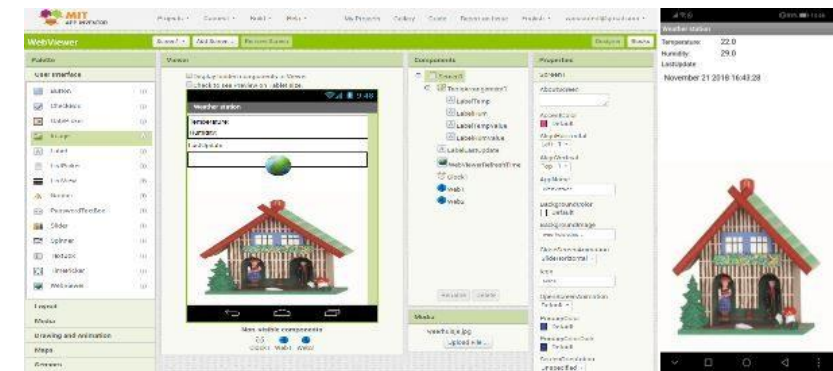
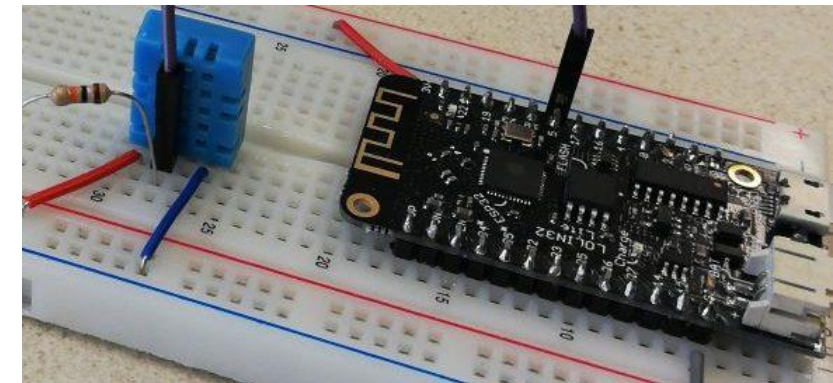
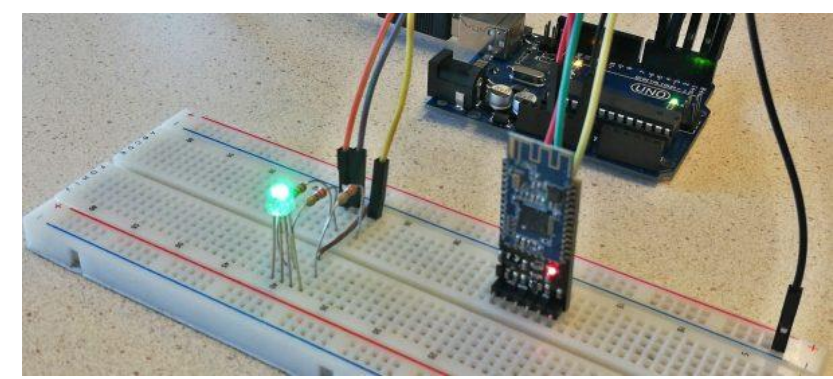
This presentation & tutorials available at: vanslooten.com/workshop

* contains breadboard, wires, Arduino Nano, power module, buttons, leds

** contains Arduino Uno EVShield, ESP Wifi module, 2 sensors, battery holder etc

LET'S DO...

- Practice building a prototype:
 - [Create a connected sensor](#)
 - [Control an RGB LED from an Android App via Bluetooth](#)
- Create a 'virtual' prototype
 - Do [Axure](#) or [Figma](#) tutorial
- Build an App
 - [Creating a connected App using App Inventor](#)
 - [Create an App with Android Studio](#)



Contact: f.vanslooten@utwente.nl, W241

This presentation & tutorials available at: vanslooten.com/workshop
Use examples and references to [tutorials](#) in this presentation

QUESTIONS?

vanslooten.com/workshop

f.vanslooten@utwente.nl
W241