LET YOUR PROTOTYPING JOURNEY BEGIN!

Industrial Design Engineering
TU Delft

INTRODUCTION

Hello, fellow prototyper!

As a design student, you are learning to navigate a diverse set of skills. While becoming an expert in every domain is nearly impossible, it is essential to understand a skill sufficiently well to talk about ideas, understand challenges and limitations, and communicate confidently with experts.

With this kit, you will develop the abilities and confidence needed to realize your vision of how technology should influence people's daily lives.

The kit provides an easy and fun gateway to start creating a great range of technology-mediated experiences. It contains a versatile collection of tools and components for you to build on as you grow your skills and confidence.

Use your imagination to find out what you can create! Be it a wearable companion device, an interactive desk lamp, or an entire connected ecosystem that brings you closer to your friends around the globe - the sky is the limit!

Have fun and be curious!

A FEW HELPFUL POINTERS

An `INPUT` is something a micro-controller can detect, such as the state of a switch, the rotation of a dial, or a microphone's signal.

An `OUTPUT` is something that can be controlled by a microcontroller, such as a light, speaker, or motor.

A `DIGITAL` signal is created by anything that assumes discrete states, such as on / off or 1 / 0.

An `ANALOG` signal is created by anything that can assume any state on a continuous spectrum, such as temperature or pressure.

A `PWM` signal is an analog-like signal created by switching a digital signal on and off very fast.

The `I²C` protocol is used to interface advanced components using two digital signals.

Core Components

01 Bitsy Expander
Expands the capabilities of your microcontroller. Provides solderless connectors, WiFi connectivity, battery management, and more.

02 Microcontroller Board
The ItsyBitsy M4. It runs code, reads and processes sensor data, and controls outputs (lights, motors, speakers, etc.)

Solderless Components

03 Servo Motor
A small motor with position control. Its range of motion is limited to approximately 180°. OUTPUT PWM

04 Touch Sensor
A capacitive switch that remains active while touched. INPUT DIGITAL

05 Piezo Buzzer
Can generate beeping sounds and melodies. OUTPUT PWM

06 Vibration Motor
Creates vibration that can be used to generate patterns comparable to cell phone alarms. OUTPUT PWM

07 Chainable RGB/w LED (x3)
A light source that can produce RGB colored and white light. Can be daisy chained. OUTPUT DIGITAL

08 Time of Flight Distance Sensor
Detects the distance of objects directly in front of the sensor. Works best with white surfaces, for a range from 30 to 1200 mm. INPUT I²C

Custom Components

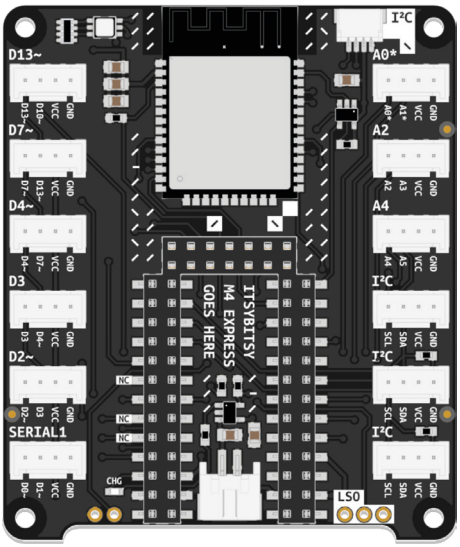
09 Tactile Switch (x2)
A push button that remains active while pressed. INPUT DIGITAL

10 Rotary Potentiometer
Changes its electrical resistance based on its rotation angle. Can detect rotation angles between 0° and 270°. INPUT ANALOG

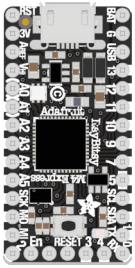
11 Photoresistor
Changes its electrical resistance in response to light intensity. Can be used to detect changes in light intensity. INPUT ANALOG

12 Tilt Switch
Detects if it has been tilted beyond a certain point. INPUT DIGITAL

13 Thermistor
Changes its electrical resistance with temperature. Can be used to estimate the current temperature and detect changes. INPUT ANALOG

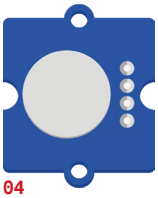


01

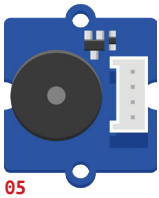


02

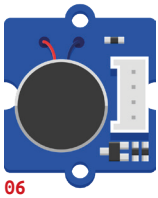
03



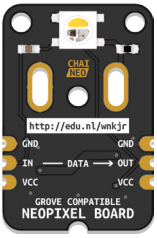
04



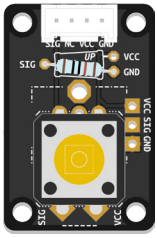
05



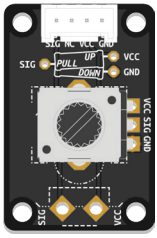
06



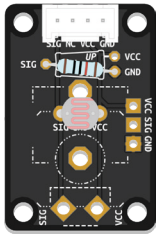
07 x3



09 x2

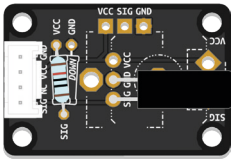


10

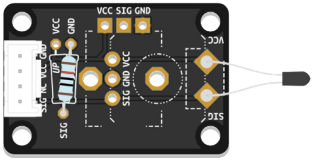


11

08



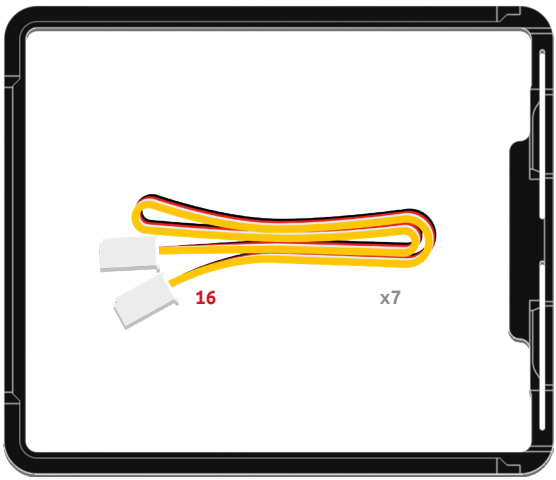
12



13

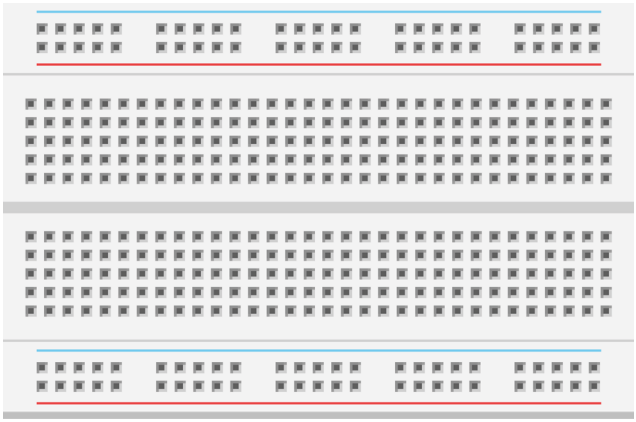
14

17



16 x7

15



18

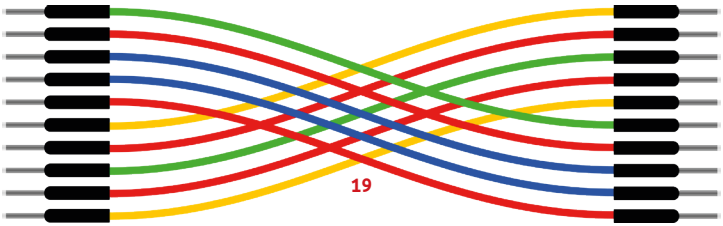
20 x5

21 x5

x2 22

x3 23

x2 24



19

Cables & Accessories

14 USB Data Cable – 100 cm
A USB-A to Micro-USB-B data cable for programming and powering the microcontroller.

15 Protective Base
Clips onto the Bitsy Expander to minimize the risk of short circuits.

16 Grove Cable (x7)
Connects components with Grove Headers to the Expander board.

17 Grove to 4-Pin Male Cable
Connects Grove components to pin headers or a breadboard.

18 Breadboard
A helpful tool for prototyping electrical circuits using components that are not pre-assembled.

19 Jumper Wire Pack
Used to prototype electrical circuits on the breadboard or to interface with pin headers.

20 Resistor 10 kΩ (x5)
Needed for the assembly of custom components, as well as for building electrical circuits on the breadboard.

21 Resistor 150 Ω (x5)
Useful for building electrical circuits on the breadboard. Suitable as a series resistor for many simple LEDs.

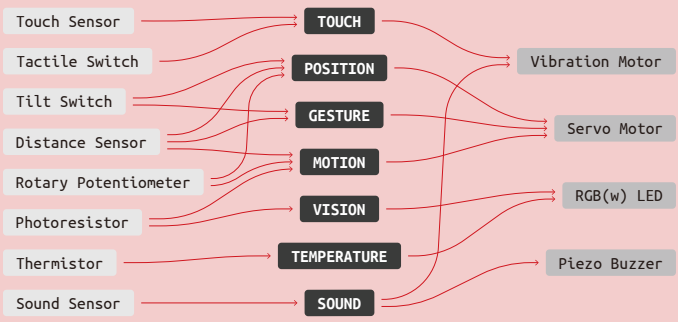
22 Button Cap (x2)
Fits the tactile switch.

23 LED Lens (x3)
Fits the Chainable RGB/w LED.

24 LED Diffuser (x2)
One matte, one frosted.

INPUT- AND OUTPUT MODALITIES ENABLED BY THIS KIT

ℹ This selection is meant as inspiration. It is not a complete list.

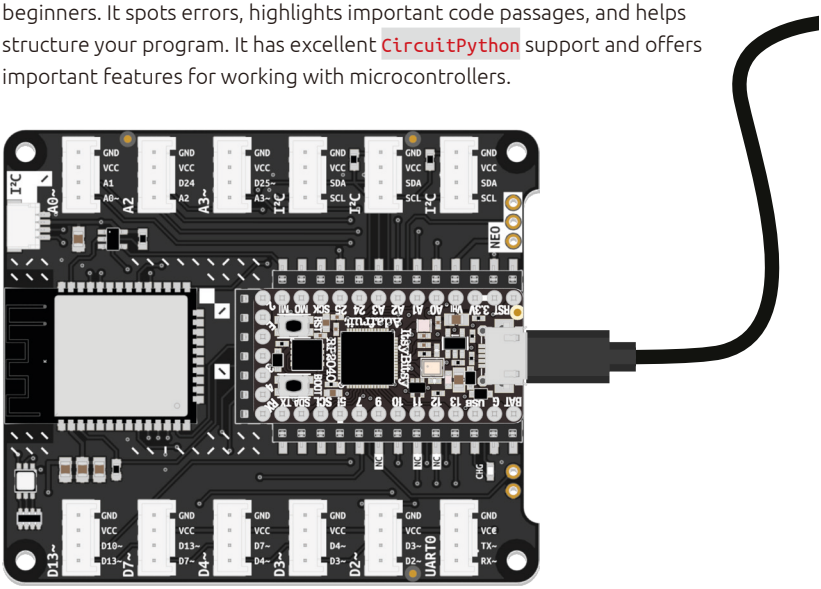


To define how your prototypes should respond to a given user interaction, you start by describing the expected behaviors as a set of rules and commands. By writing these directions down using a programming language your microcontroller can understand, you instruct it to enact the desired behaviors. The ItsyBitsy microcontroller included in your kit works best with a beginner-friendly programming language called `CircuitPython`.

Let's go over the tools and equipment you need to get started:

ItsyBitsy + Expander
The ItsyBitsy microcontroller and its Expander form the hardware platform you will be using. When connected to your computer using the included USB data cable, the ItsyBitsy will appear as a storage device called `CIRCUITPY`.

Your Computer + Mu Editor
You could, in theory, use any text editor to write programs for your microcontroller. In practice, you should always use a code editor, as it will simplify things greatly. The `Mu Editor` is free, open-source, and great for beginners. It spots errors, highlights important code passages, and helps structure your program. It has excellent `CircuitPython` support and offers important features for working with microcontrollers.



Installing Mu Editor

Before you write your first program, you need to complete one last step:

1. Visit ***codewith.mu*** and click the green Download button.
2. Select the correct link for your operating system (Windows or Mac OS).
3. Once the download is complete, launch the installer and follow the instructions on the screen.
4. On its first run, Mu will ask you to select a mode. Choose CircuitPython.

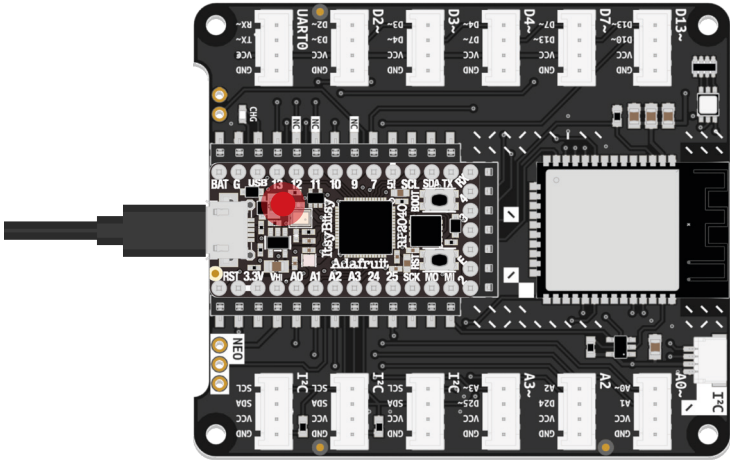
Should you run into issues during the installation process, you can find more detailed instructions on the website next to the download button for your operating system.

Tutorial 2: LET THERE BE LIGHT!

This tutorial will teach you how to use your microcontroller to make a simple LED blink. To help you get started, we pre-loaded your ItsyBitsy with a bit of code.

Part 1: Load the program code using Mu Editor

1. Using the USB cable included in the kit, connect the ItsyBitsy to your computer. It will appear in your computer's file manager as a storage device called `CIRCUITPY`.
2. Open Mu Editor. Click the Load button in the toolbar at the top of the program window and navigate to the `CIRCUITPY` drive. Find the file named `code.py` and open it.
3. Your microcontroller will execute any code stored in that file. Briefly examine the example code and see if you understand some parts. In the next step, we will take a closer look at how it works and how to tweak it.



Once you store changes in `code.py` to the `CIRCUITPY` drive, your ItsyBitsy will begin running the new code immediately. Changes will persist even after disconnecting the ItsyBitsy from your computer.

Part 2: Change the way your ItsyBitsy behaves

For now, let's ignore the first half of the code and go directly to the passage that begins with `while True:`

The `while` statement creates a `loop`. While its condition is met, the code contained within it will keep looping. The lines of code that follow are indented, indicating that they are inside the loop. Given that the condition to meet is set to `True` (which will never be False), the code will loop forever.

1. The code in the loop makes an LED built into your ItsyBitsy blink. Can you make out how the blinking behavior is created?
2. Experiment with the blinking frequency of the LED. Change the values defined in `time.sleep(1.0)`, then press the `Save` button in Mu Editor's toolbar to store the changes to your code and see their effect.
3. Click the `Serial` button in Mu's toolbar to open the `Serial Monitor`. This vital tool lets you see messages printed by your program. Can you figure out how to change the content of the messages appearing in the serial monitor? Remember to save your changes to see their effect.

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT

while True:
    time.sleep(1.0)
    led.value = True
    print("LED is on")

    time.sleep(1.0)
    led.value = False
    print("LED is off")
```

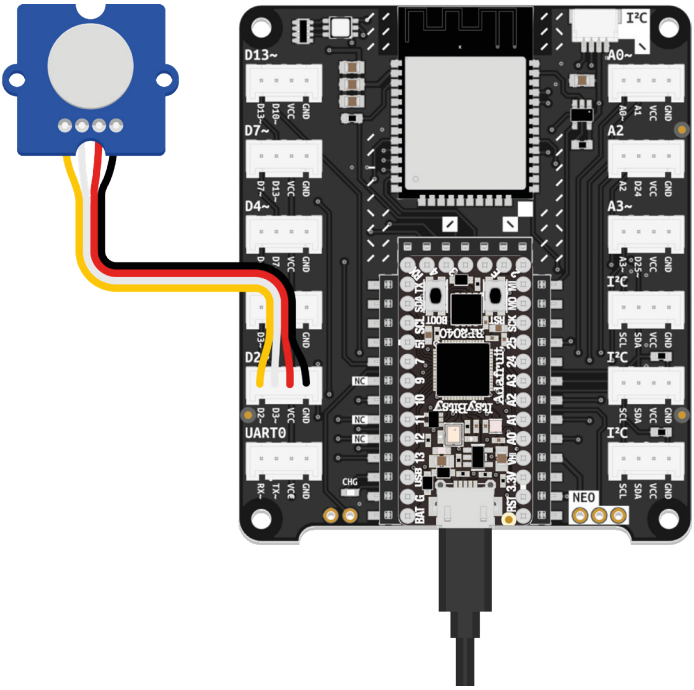
Tutorial 3: ADDING INPUTS AND OUTPUTS

In the previous tutorial, you changed the behavior of a blinking LED. Here, you will learn how to build and program your first complete interactive prototype.

You will use a sensor from your kit (an `INPUT` component) to detect an action performed by a user. Once detected, this will trigger a change in your prototype's behavior. You will use an actuator (an `OUTPUT` component) to create this behavior.

Designing this interaction provides a simple template for using the different component types in your kit. You will later be able to use this template to devise more intricate interaction choreographies.

This tutorial begins by walking you through using a `Touch Sensor` to register a user's touch. Later, you will learn how to add a `Vibration Motor` to create vibrations for as long as the sensor is triggered.



Part 1: Add an input component to your prototype

The code in this tutorial is similar to the program you saw before. This time, you will write it yourself to better understand what you are doing.

1. Begin by connecting your ItsyBitsy to the computer. We want to start this tutorial with an empty `code.py` file on your `CIRCUITPY` drive, so you need to delete or rename the existing one. For instance, you can use your computer's file manager to give the old file from the last tutorial the name `blink_code.py`. Remember that your microcontroller will only execute code stored in a file named `code.py`.

2. Temporarily disconnect your ItsyBitsy from the computer. Use a Grove cable to connect the Touch Sensor to the pin connector labeled `D2` on the Expander Board, as shown in the illustration on the left.

3. Reconnect your ItsyBitsy, then open Mu Editor. Click the `New` button in the toolbar at the top to create a new file. Then click `Save` to store your new file on the `CIRCUITPY` drive, naming it `code.py`.

4. Follow along with the code example on the next page by beginning your program with `import` statements. These inform your board about additional instruction sets your code needs to work. These instruction sets are called `libraries` or `modules`. You need to import three modules: `board` tells your program about the pins available on your microcontroller and their names. `digitalio` contains functions necessary for working with digital inputs and outputs. `time` allows your code to use timing-related functionality.

`Libraries` and `modules` contain code written by other people to fulfill specific tasks. Core modules, such as `board`, `digitalio`, and `time` provide functionality essential to working with your board. Therefore, they are already included in CircuitPython. In some cases, you may need to download additional libraries to add functionality, such as drivers for specific sensors, to your code. You can learn more about that subject by reading the chapter on CircuitPython Libraries in Adafruit's guide at *learn.adafruit.com/welcome-to-circuitpython/circuitpython-libraries*.

5. To make the Touch Sensor you connected in step 2 work, you first need to create a container to store the data coming from the sensor. Such a container is called a `variable`. Variables are created by assigning a name to something: From now on, `sensor` will store a `digitalio` object attached to pin `D2`. In the following line of code, you define the value stored in `sensor` to be read from an `INPUT`, not written to an output.

6. As before, you end your program with a `while` loop whose condition is set to `True`. Inside this endlessly repeating loop, you use a `print` statement to display the `sensor.value` in the `Serial Monitor` each time the code repeats. Using a sleep timer, you slow down the speed of the loop so as not to overwhelm Mu's Serial Monitor with more messages than it can handle.

```
import board
import digitalio
import time

sensor = digitalio.DigitalInOut(board.D2)
sensor.direction = digitalio.Direction.INPUT

while True:
    print(sensor.value)
    time.sleep(0.1)
```

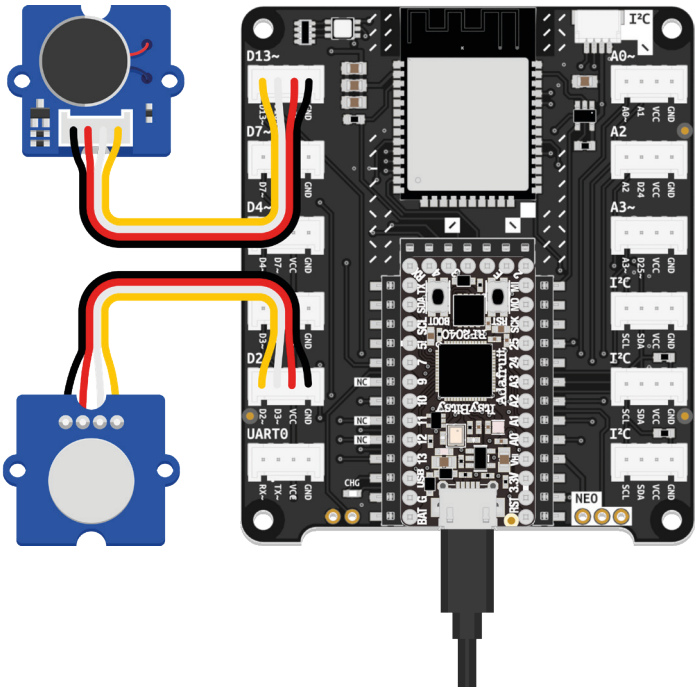
It is advisable to regularly back up the code stored on the `CIRCUITPY` drive to your computer. That way, you have something to fall back on should a memory loss occur or your ItsyBitsy is misplaced.

Part 2: Add an output component to your prototype

Now that your prototype can register a user input, it is time to tie it together by adding your output component.

1. As with the Touch Sensor, begin by connecting your Vibration Motor using a Grove cable. Connect it to pin `D13`, as shown in the illustration below.

2. As before, you need a variable to hold data to make your component work. Make sure to give it a sensible name, such as `motor`. This time, instead of reading a value from a sensor, you will use the variable to write data to an actuator. The statements needed for the vibration motor will look very similar to the ones you wrote for the touch sensor, with one critical difference: As shown in the following example, the `digitalio.Direction` is set to `OUTPUT`.



3. As you recall, the goal is to make the motor vibrate for as long as the sensor registers a touch. If the sensor is not touched, no vibration should be created. CircuitPython provides a handy way to translate this idea into code using `if...else statements`. An `if statement` executes a block of code only if a specified condition is `True`. An `else clause` can be added to run another set of instructions should the condition be false.

4. You can use this knowledge to turn the motor on and off, depending on the current state of the touch sensor. If `sensor.value is True:`, the motor can be turned on with `motor.value = True`. Else, the motor needs to be turned off again. Remember to save your code to see it in action.

5. Try experimenting with the code to change the behavior of your prototype. See how programmed behavior is affected by using the Piezo Buzzer as an actuator instead of the motor.

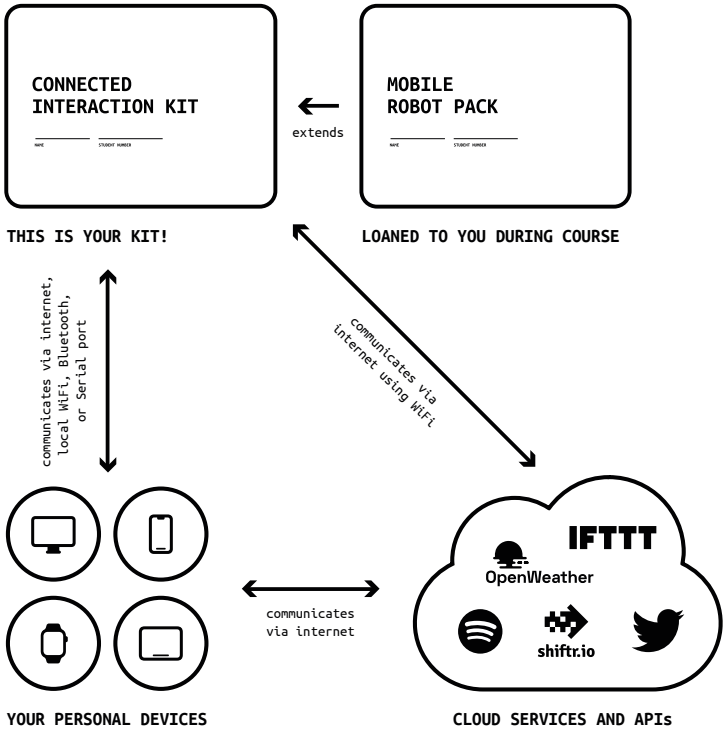
```
import board
import digitalio
import time

sensor = digitalio.DigitalInOut(board.D2)
sensor.direction = digitalio.Direction.INPUT

motor = digitalio.DigitalInOut(board.D13)
motor.direction = digitalio.Direction.OUTPUT


while True:
    print(sensor.value)
    if sensor.value is True:
        motor.value = True
    else:
        motor.value = False
    time.sleep(0.1)
```

SPACE FOR THOUGHTS AND IDEAS

You acquired the Basic Drawing and Modelling Pack at the beginning of your bachelor's program, giving you the essential tools to draw and model your ideas. The Connected Interaction Kit now gives you the tools to breathe life into those ideas.

Both these kits were designed to complement a collection of tools already at your disposal. Together with your computer or tablet, mobile phone, and countless online services, they form an ecosystem to support your journey into the world of connected interactive prototyping.

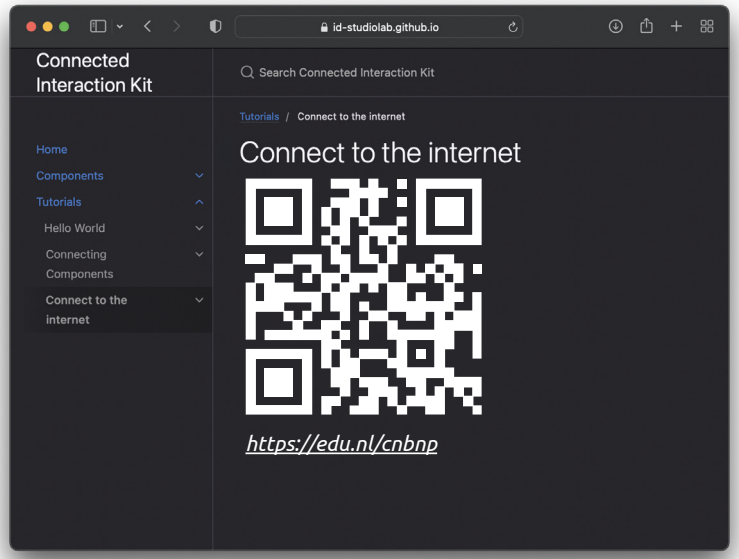
The ecosystem also includes extension packs designed for specific courses, such as the Mobile Robot Pack loaned to you during the Product Engineering course.



Tutorial 4: CONNECTING TO THE INTERNET

This tutorial will teach you how to unlock the connected aspects of your Connected Interaction Kit. Different from the previous tutorials, you will not find this one in the booklet. Instead, you can find it, along with more resources, in the documentation attached to the `GitHub repository` for this kit.

To access it, visit the link below or scan the QR code.



USEFUL LINKS AND RESOURCES

This page contains a variety of helpful links and resources for your projects. If you need help finding what you're looking for, feel free to ask your coaches for additional support.

Additional Documentation & Resources

The following links provide access to online resources related to the Connected Interaction Kit, including detailed component descriptions, additional tutorials, and 3D printable add-ons.

https://id-studiolab.github.io/Connected-Interaction-Kit
https://learn.adafruit.com/adafruit-itsybitsy-rp2040

Where To Buy Components?

If you need a specific component that is not included in the kit, the following online stores ship to the Netherlands and may have what you need.

https://www.kiwi-electronics.com/
https://www.sossolutions.nl/
https://www.reichelt.nl/
https://www.conrad.nl/

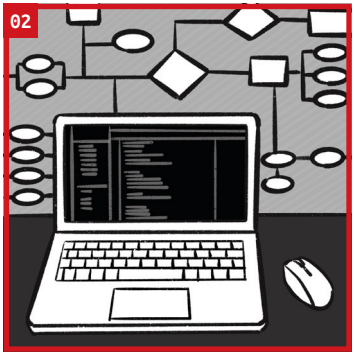
Useful Knowledge

These links provide valuable information on topics such as getting started with CircuitPython and the basics of electronics.

https://learn.adafruit.com/welcome-to-circuitpython
https://learn.sparkfun.com/tutorials/voltage-current-resistance-and-ohms-law

ℹ In addition to the ready-made solderless Grove components, the Connected Interaction Kit also provides several `Custom Components`. To experiment with or use one of them with the code examples in this booklet, you must assemble them first. Detailed assembly instructions and a soldering tutorial can be found by visiting the abovementioned documentation under *https://edu.nl/a3au7*.

SPACE FOR THOUGHTS AND IDEAS

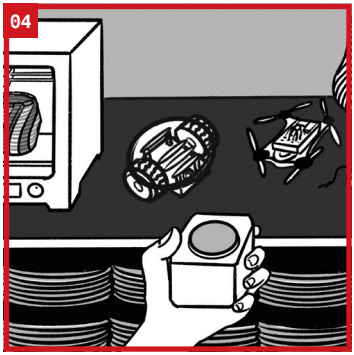


01 Designing Product Services (DP2)

In this course, you will learn to apply lean startup methodologies to develop an innovative product-service system that considers the needs of its stakeholders. Internet-connected home appliances form a significant portion of this innovation space, including intelligent lighting systems and automated cat feeders. You can use the Connected Interaction Kit to create working prototypes of your ideas, evaluate their feasibility, and test them with peers.

02 Digital Product Development

This course introduces you to the design of digital products and services. Understanding the basics of software development is central to the course, as it differs significantly from what you already know about designing tangible products. You will learn Python programming basics and how organizations handle the software development process. You will also explore different networking technologies and understand the importance of considering privacy and security in the design process. Combined with the knowledge introduced by the Connected Interaction Kit, this course will enable you to tackle ambitious integrated projects in the future.



03 Digital Interfaces

In this course, you will learn about Human Computer Interaction (HCI) from various perspectives. Firstly, you will examine the design of interfaces from a technological standpoint. Secondly, you will consider the role of the designer in creating static and dynamic interfaces. Finally, you will explore the interactions and experiences users have with an interface. Throughout the course, you will use the digital prototyping ecosystem to participate in several workshops where you will build and analyze interfaces with fellow students and receive guidance from your coaches.

04 Product Engineering

In this course, you will build a robot that can move and perform a simple task in order to understand how to design products that use motion. You will learn how to calculate the movement of objects, design with mechanics and electronics, and understand the forces at play. The Mobile Robot Pack will extend your Connected Interaction Kit's digital prototyping tools for the duration of the course.

05 Form and Senses

This course teaches you how to incorporate multi-sensory experiences into your design process. You can use tools in the digital prototyping ecosystem to explore how sound, texture, and light design can impact a user's experiences. The Connected Interaction Kit offers a diverse range of components that allow you to explore various multi-sensory input and output forms.

06 Interactive Environments Minor

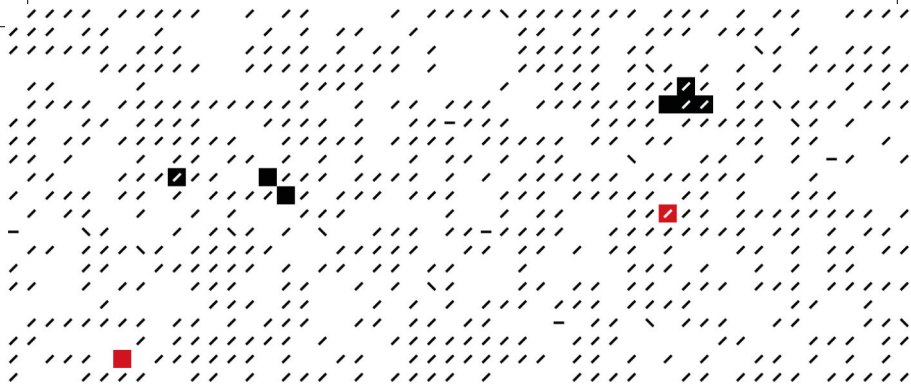
This course teaches you how to design interactions within architectural spaces. You will learn how to consider the interactions between technology and people in these environments. You will develop skills individually and work with a team to create a full-scale prototype. Using the Connected Interaction Kit's digital prototyping ecosystem as a foundation, you will begin to incorporate more advanced technologies to enhance the design of your experiences.

07 Bachelor 's End Project

In your bachelor's end project, you will design a solution to a real-world challenge to demonstrate that you are able to apply the skills and knowledge you have gained during your studies. You can use the prototyping ecosystem of the Connected Interaction Kit to create prototypes that test your design ideas and communicate them to stakeholders in a way that generates enthusiasm for the concept.

08 Build Whatever You Want!

The Connected Interaction Kit's digital prototyping ecosystem is a powerful and enjoyable tool that helps you build the skills and confidence to develop interactive projects. You can use it to start with simple prototypes and gradually improve your understanding of technology, or you can immediately tackle more complex ideas. In addition to its educational value, the kit allows you to bring fun and functionality to your daily life. It truly shines when you use it to explore your ideas and inspirations, creating things you enjoy or that enhance your daily routines.



TU Delft

Authors
Adriaan Bernstein, Aadjan van der Helm, Frederik Ueberschär

Layout and Graphic Design
Adriaan Bernstein, Frederik Ueberschär, Yeun Kim

Acknowledgments
Martin Havranek, Lorenzo Romagnoli, Jack Eichenlaub