Aalto University
School of Arts, Design
and Architecture

Programming 2:
Basics of programming with Arduino

Wearable technology and functional wear

Antti Salovaara

What you'll learn today

More interactiveness to Arduino project

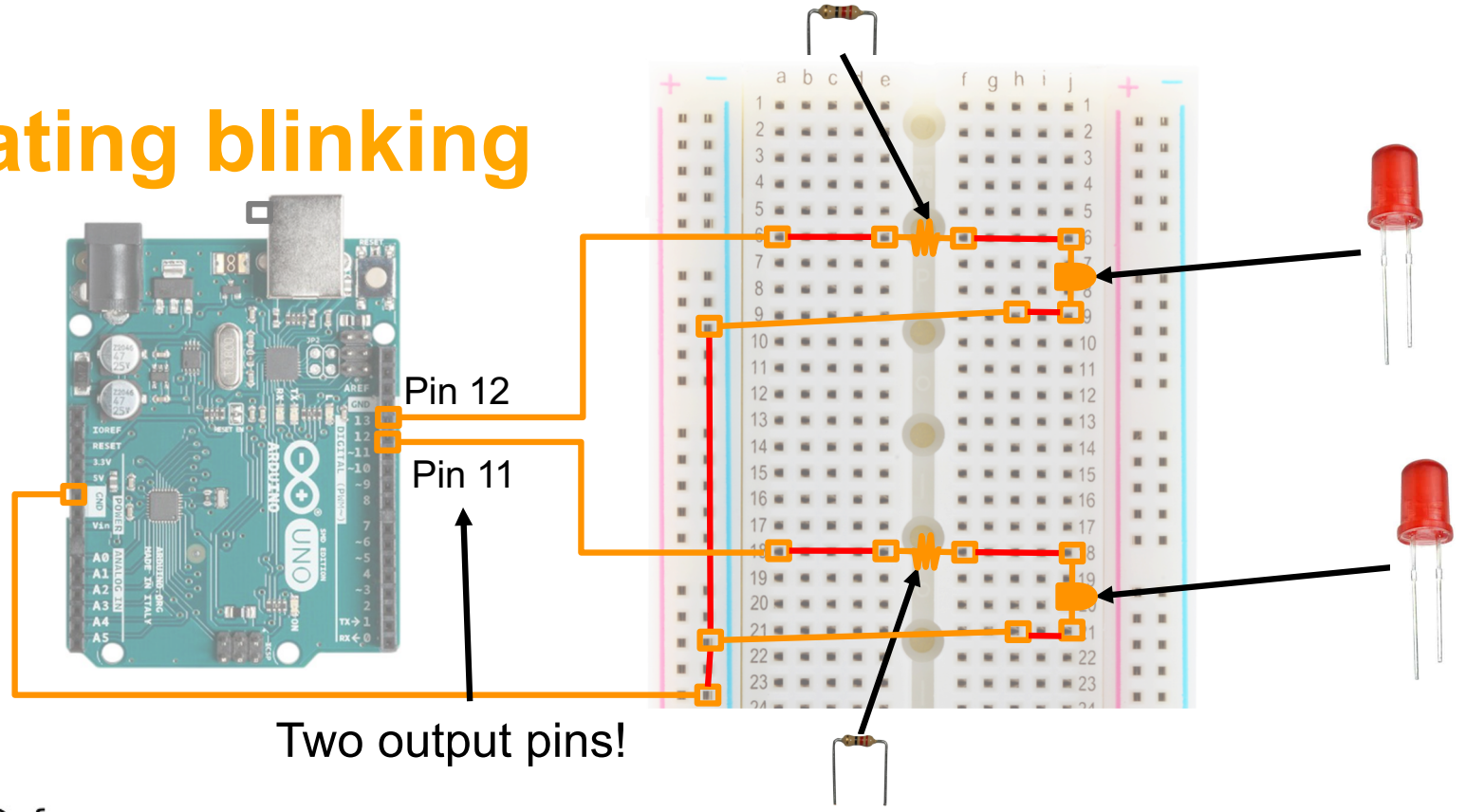
- Using more variables and if-else blocks

How to write good code

- Writing own functions to modularize the code
- Commenting
- Avoiding "hard coding"
- Useful conventions

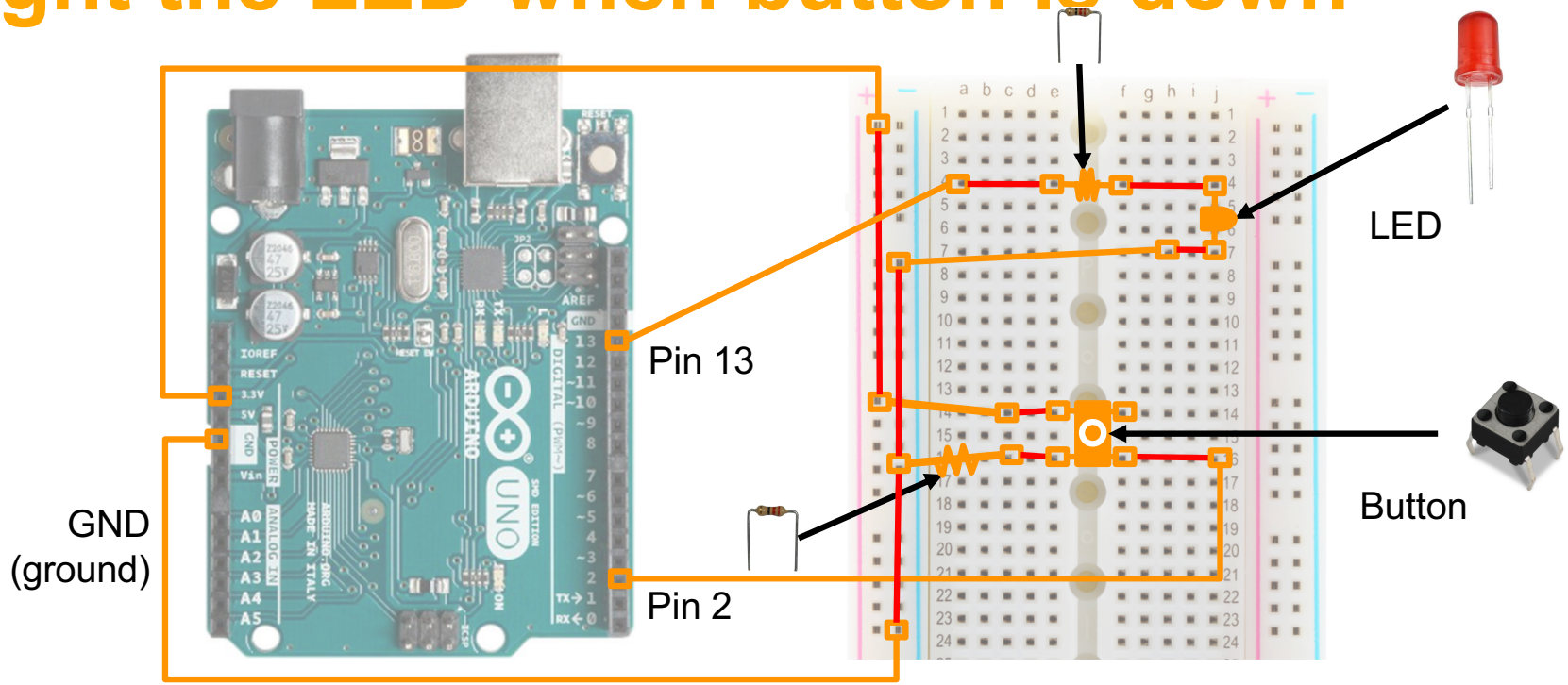
Repetition:
How far did we reach last time?

Alternating blinking




```
void setup() {
  // put your setup code here, to run once:
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  delay(1000);

  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  delay(1000);
}
```

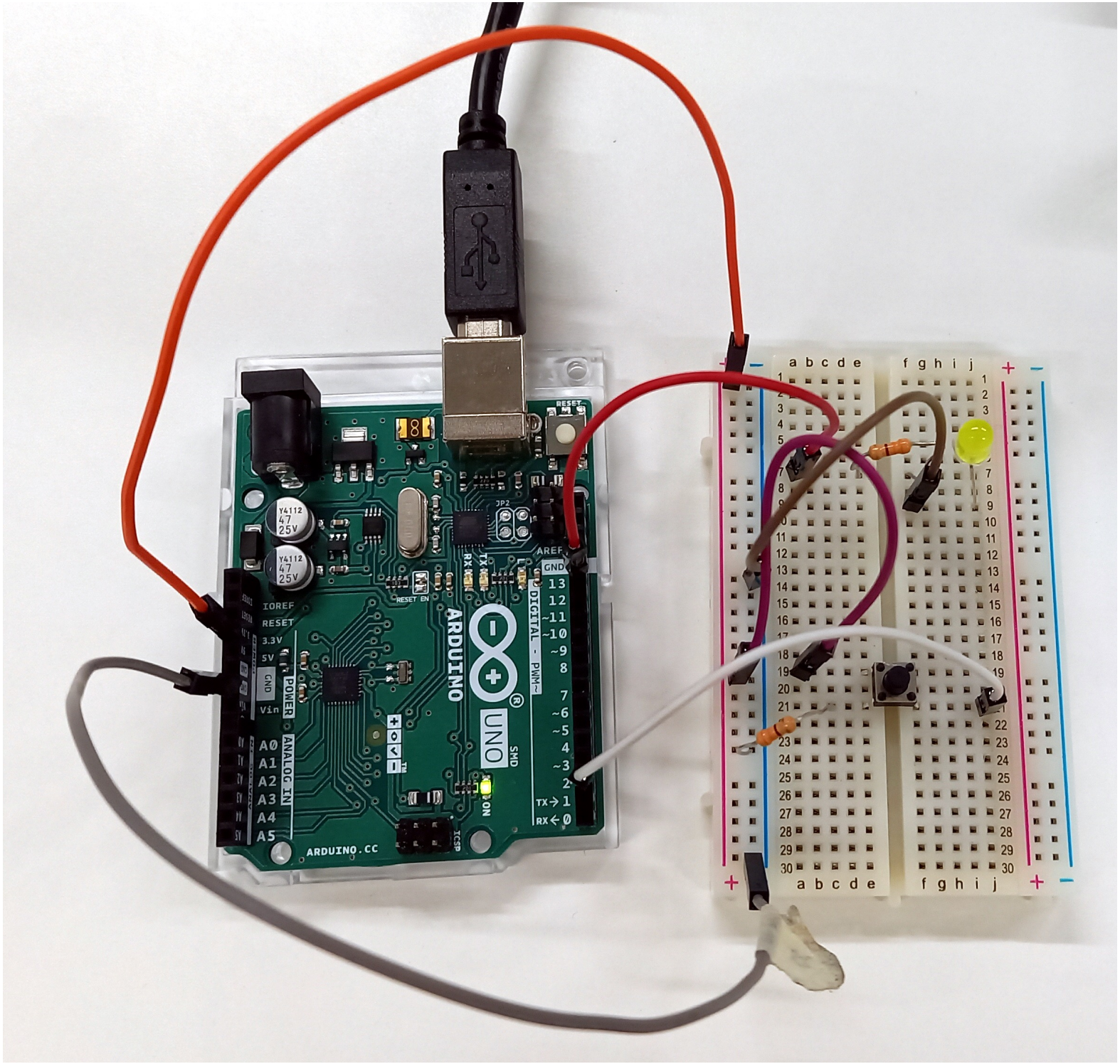
Light the LED when button is down




```
void setup() {
  pinMode(2,INPUT);       // detect button press from pin 2
  pinMode(13,OUTPUT);     // control LED from pin 13

  digitalWrite(13,LOW);   // start by having the LED off
}

void loop() {

  // read the current button state and store the result in a variable:
  int buttonState = digitalRead(2);

  // do different actions based on what state the button has
  if (buttonState == HIGH) {
    // HIGH means that the button is pressed => turn on the LED
    digitalWrite(13,HIGH);
  }
  else {
    // The other option: the button is not pressed => turn the LED off
    digitalWrite(13,LOW);
  }
}
```

Photograph of this setup is in the next slide

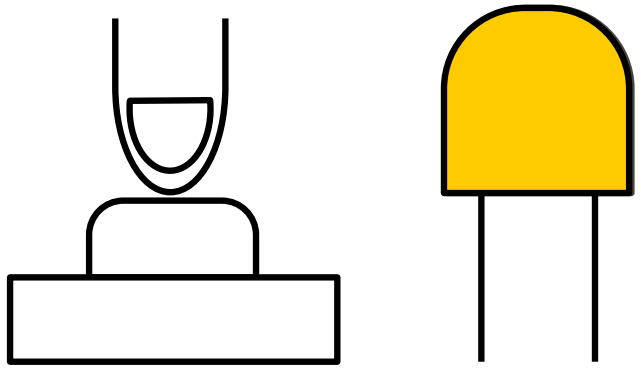
button-detector.ino

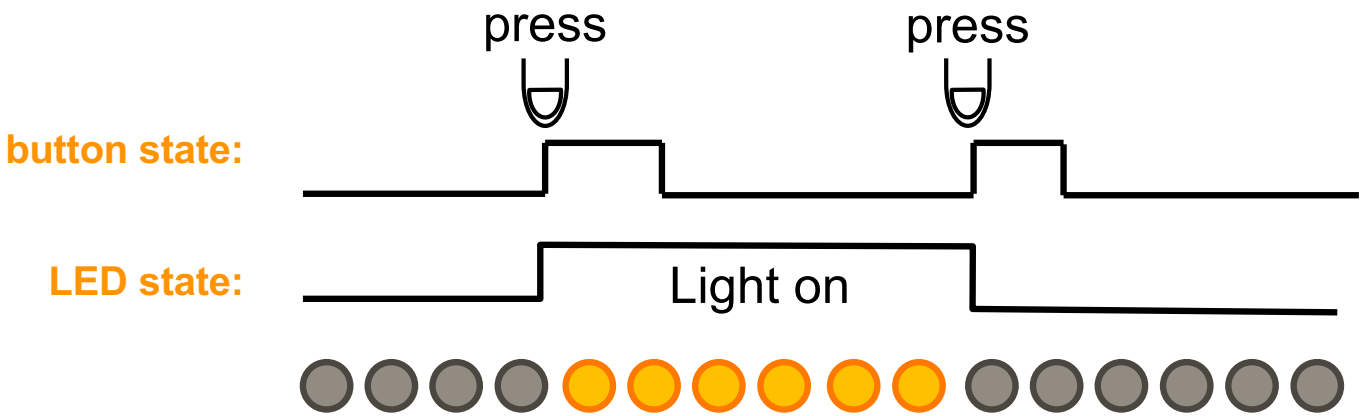

ARDUINO
UNO
SMD
ARDUINO.CC
POWER
ANALOG IN
DIGITAL - PWM~
IOREF
RESET
3.3V
5V
GND
Vin
A0
A1
A2
A3
A4
A5
AREF
RESET EN
ICSP
ON
TX→1
RX←0
a b c d e
f g h i j

New stuff:
A button that keeps the light on after a press

Button project 2:

"Toggle" the LED on/off on each button press

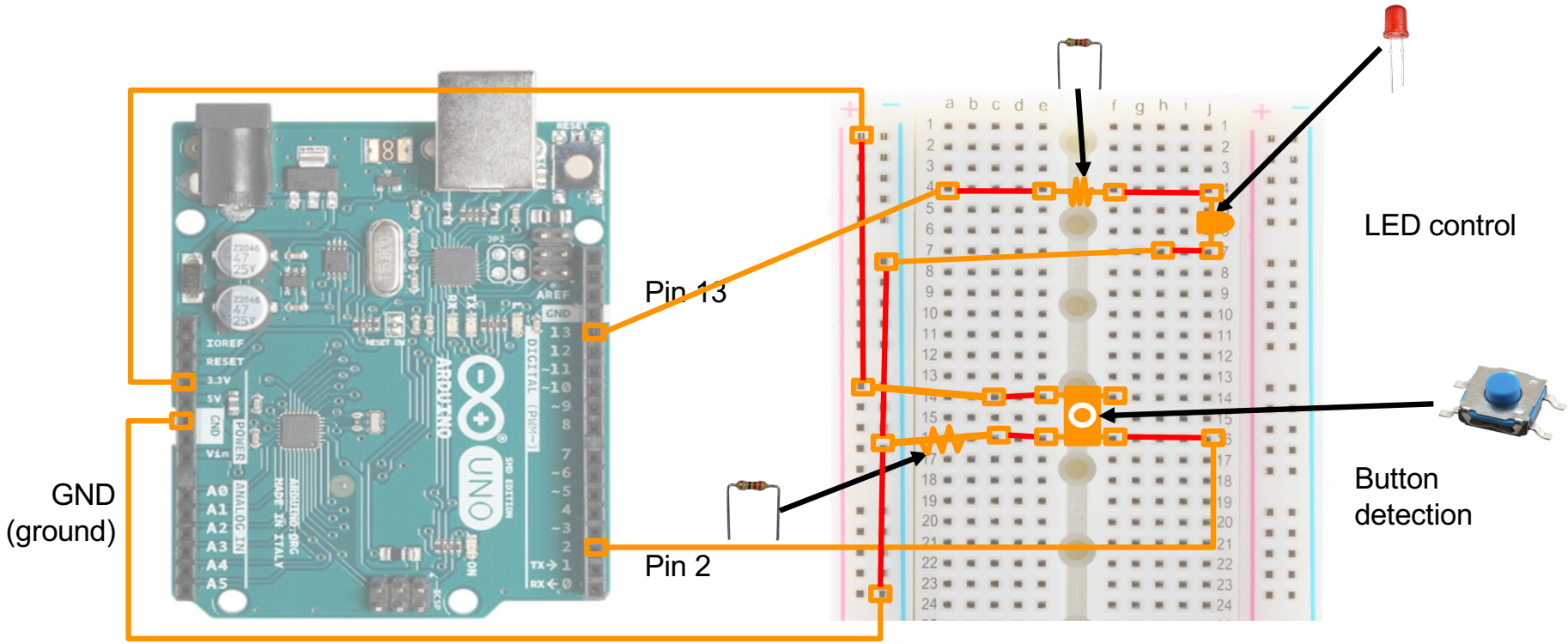




Toggle the LED on/off on every button press

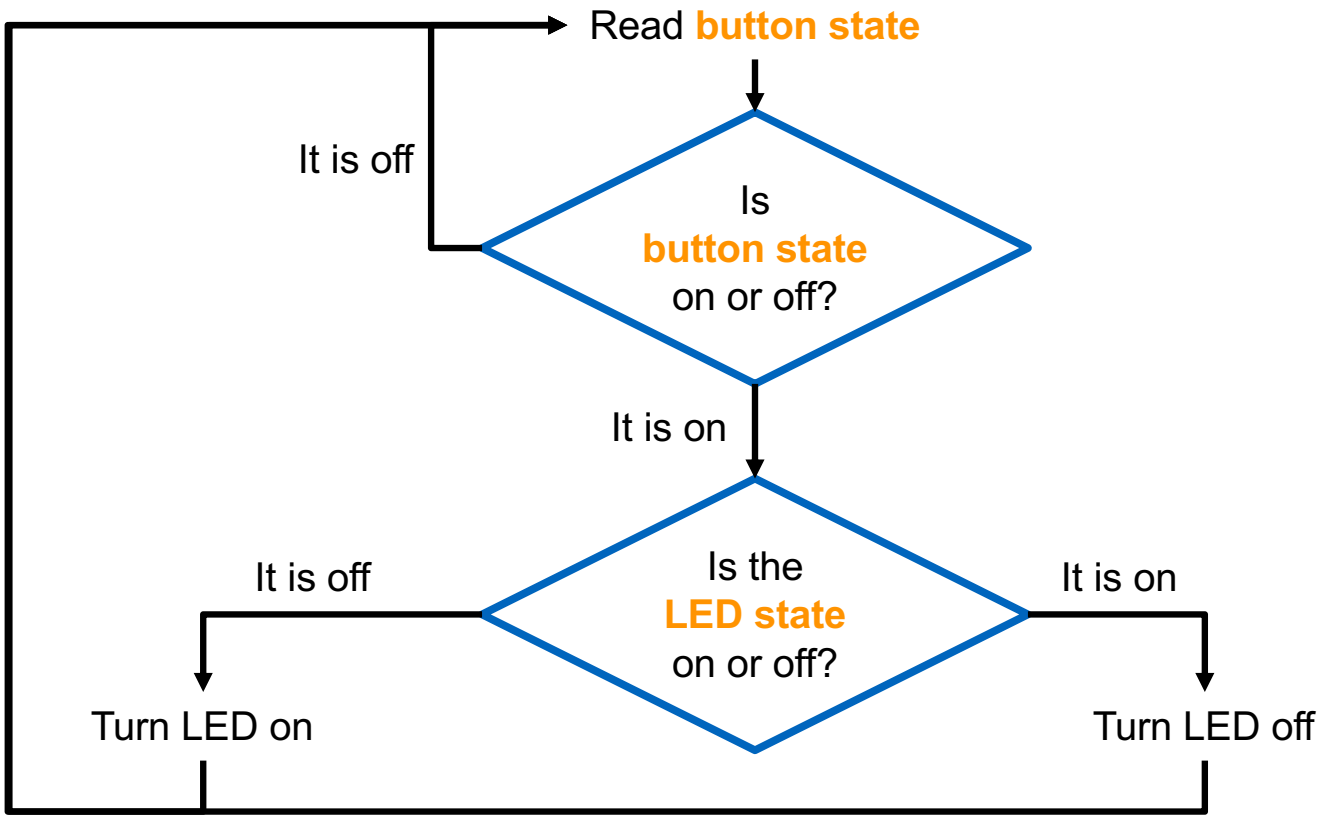
Same wiring as in the previous button project

But more programming: See the next pages

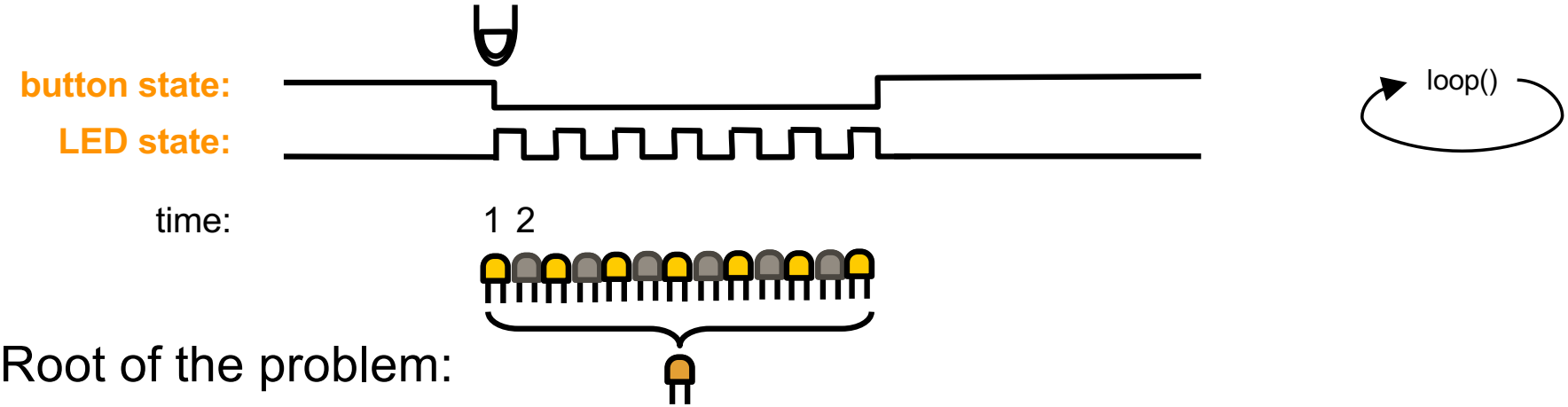


A simple plan for loop()'s logic

(…that does not work yet – wait for next page)



The problem with the simple logic



Root of the problem:

loop() runs hundreds of times every second, but a finger's press lasts several runs of that loop.

Time 1: Button is pressed and LED is off. Therefore we switch the LED on.

Time 2: Button is still pressed and now LED is on. Therefore we switch the LED off.

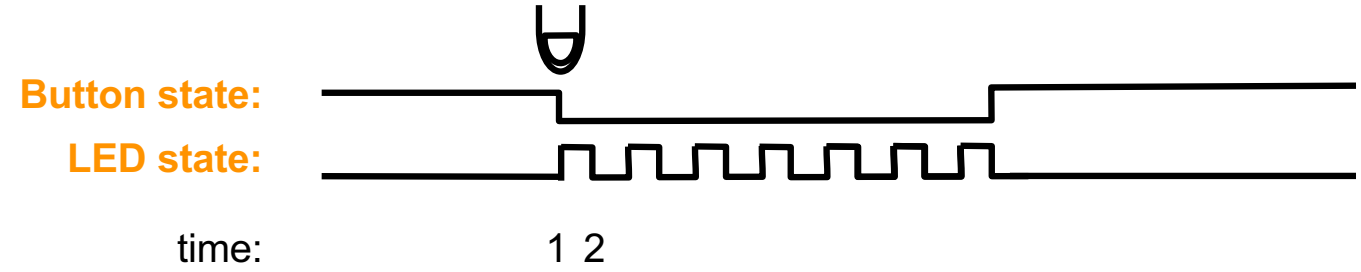
What happens:

If we change the LED state every time when we notice that button is down, we change the LED on/off on every run of the loop()

Result: we have a very quickly blinking LED and it looks like it's on all the time

How the problem can be solved

Non-working solution acted like this:

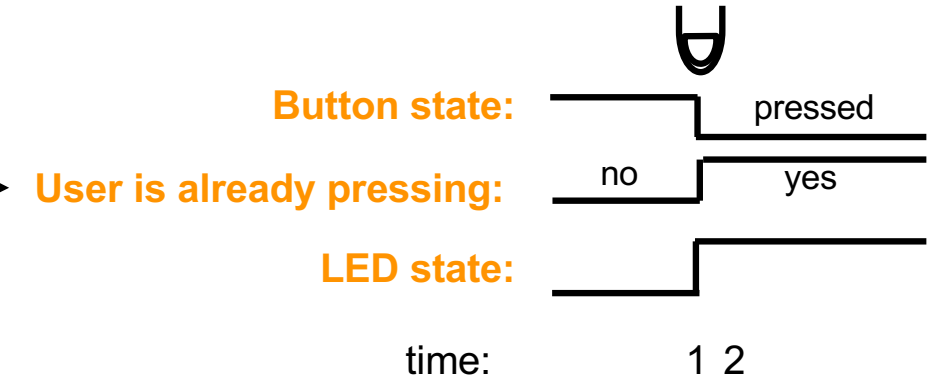


Solution:

We want to know if user is already pressing the button.

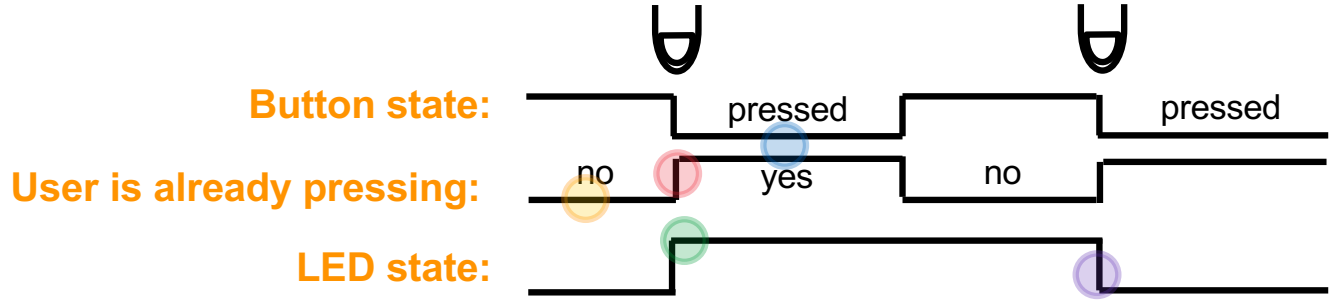
Then at Time 2 we know that we don't do anything.

In this new variable, we remember if the user alread presses the button →

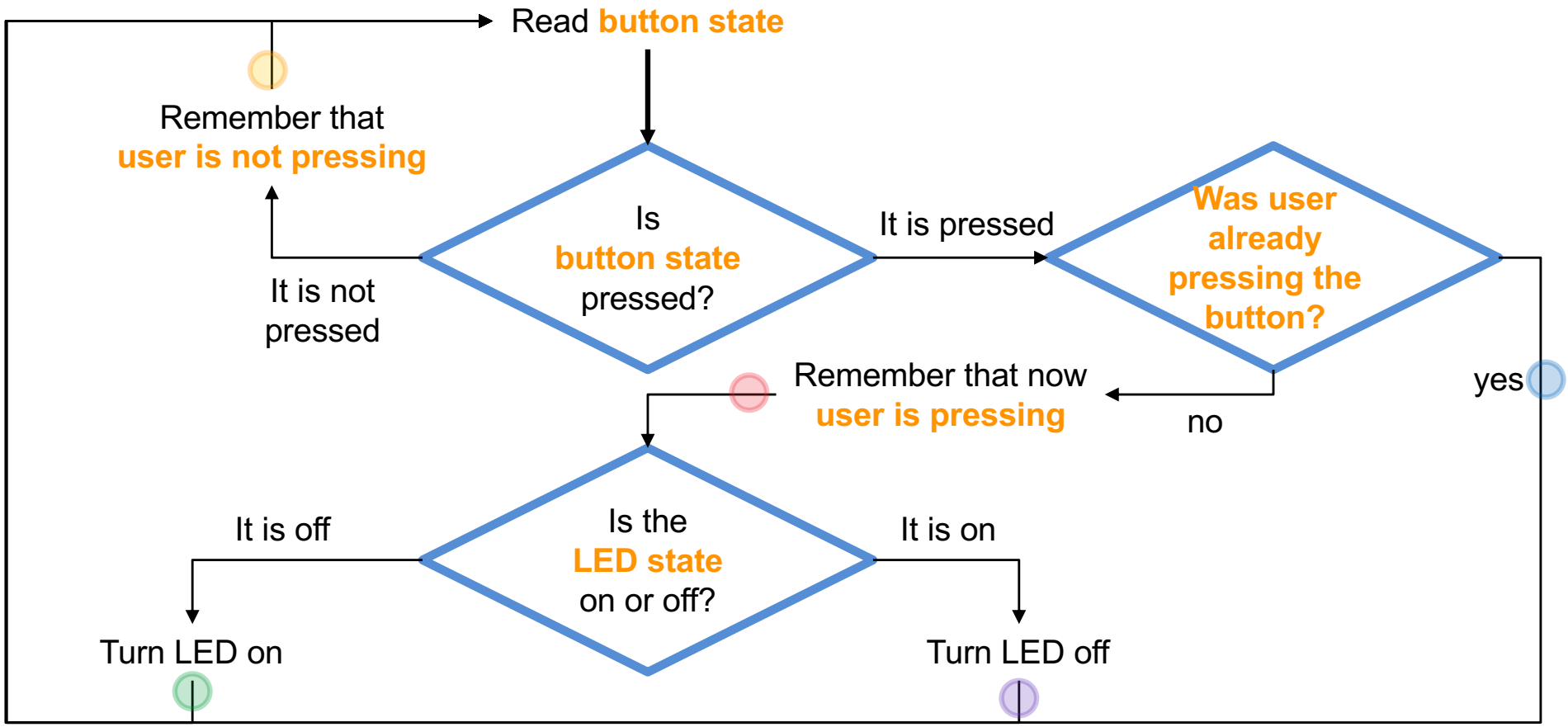


We change LED only if button is pressed and user is not already pressing the button

The working solution



We change the LED on/off only 1) when button is pressed and 2) user was not already pressing it.

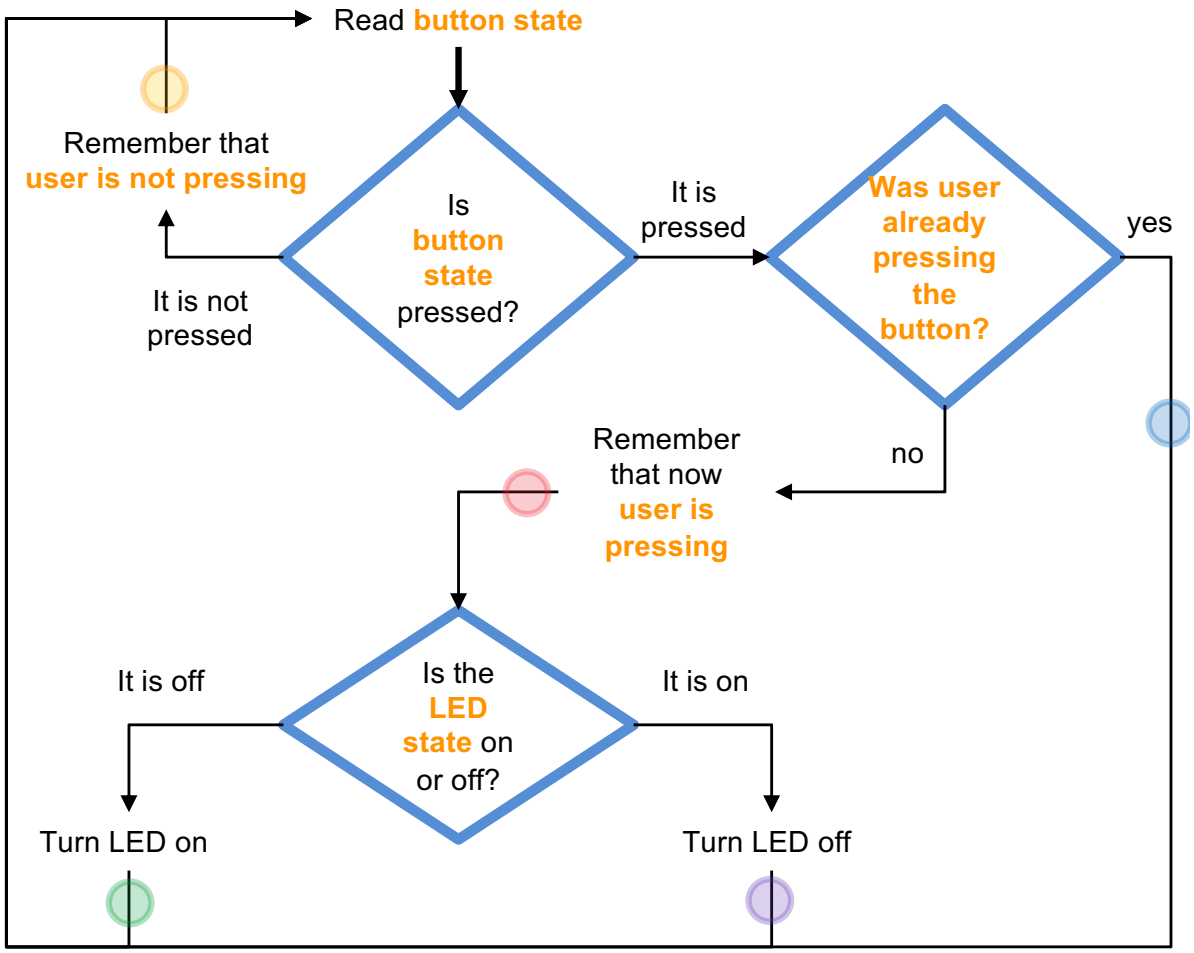


Let’s program the toggling button

Create a new empty project in Arduino for this

```
void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    // if user is not pressing the button now,
    // tell that for later loops:
    userAlreadyPressesButton = false;
  }
  else {

    if (userAlreadyPressesButton == false) {
      // This is the situation where the user
      // has just now pressed down the button,
      // and was not doing it earlier.

      // tell to later loops that user is now
      // pressing the button:
      userAlreadyPressesButton = true;

      // Toggling:
      // if LED was not on, turn it on;
      // if LED was on, turn it off:
      if (ledIsOn == false) {
        digitalWrite(13,HIGH);
        ledIsOn = true;
      } else {
        digitalWrite(13,LOW);
        ledIsOn = false;
      }
    }
    else {
      // do nothing because this just means that
      // user has not lifted the finger from the
      // button yet.
    }
  }
}
```

loop()



button-toggler.ino

setup()

```
bool ledIsOn;
bool userAlreadyPressesButton;

void setup() {
  // we'll detect button press from pin 2:
  pinMode(2,INPUT);

  // we'll control LED from pin 13:
  pinMode(13,OUTPUT);

  // we start by having LED off:
  digitalWrite(13,LOW);

  // we remember this LED state in our variable
  ledIsOn = false;

  // we start from state where user is
  // not pressing the button:
  userAlreadyPressesButton = false;
}
```

"bool" means that these variables' only values can be "true" and "false"

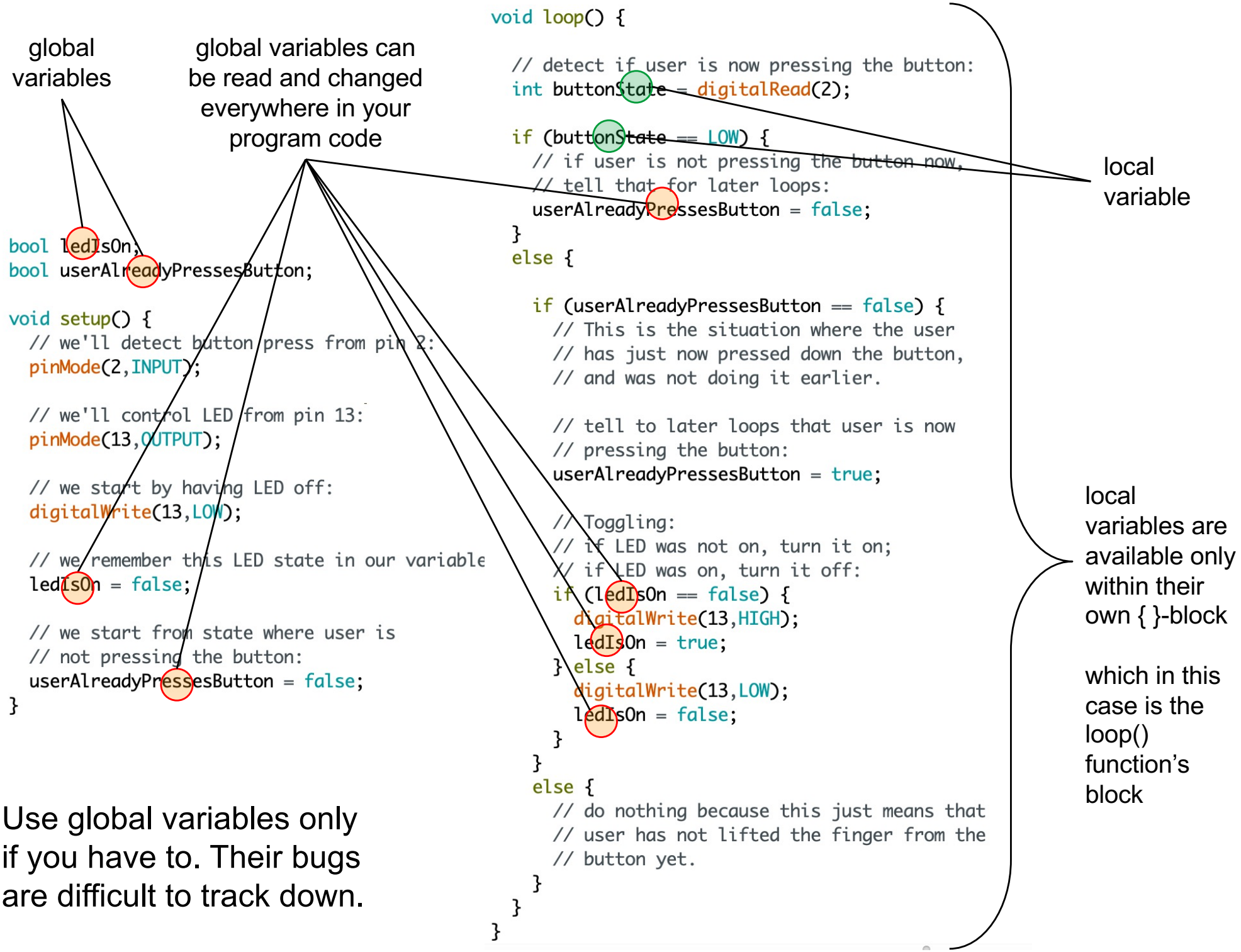
Let's now try this out, then examine the code in more depth

button-toggler.ino

All in one screen

```
bool ledIsOn;
bool userAlreadyPressesButton;

void setup() {
  // we'll detect button press from pin 2:
  pinMode(2,INPUT);

  // we'll control LED from pin 13:
  pinMode(13,OUTPUT);

  // we start by having LED off:
  digitalWrite(13,LOW);

  // we remember this LED state in our variable
  ledIsOn = false;

  // we start from state where user is
  // not pressing the button:
  userAlreadyPressesButton = false;
}
```

```
void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    // if user is not pressing the button now,
    // tell that for later loops:
    userAlreadyPressesButton = false;
  }
  else {

    if (userAlreadyPressesButton == false) {
      // This is the situation where the user
      // has just now pressed down the button,
      // and was not doing it earlier.

      // tell to later loops that user is now
      // pressing the button:
      userAlreadyPressesButton = true;

      // Toggling:
      // if LED was not on, turn it on;
      // if LED was on, turn it off:
      if (ledIsOn == false) {
        digitalWrite(13,HIGH);
        ledIsOn = true;
      } else {
        digitalWrite(13,LOW);
        ledIsOn = false;
      }
    }
    else {
      // do nothing because this just means that
      // user has not lifted the finger from the
      // button yet.
    }
  }
}
```




```
bool ledIsOn;
bool userAlreadyPressesButton;

void setup() {
  // we'll detect button press from pin 2:
  pinMode(2,INPUT);

  // we'll control LED from pin 13:
  pinMode(13,OUTPUT);

  // we start by having LED off:
  digitalWrite(13,LOW);

  // we remember this LED state in our variable
  ledIsOn = false;

  // we start from state where user is
  // not pressing the button:
  userAlreadyPressesButton = false;
}

void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    // if user is not pressing the button now,
    // tell that for later loops:
    userAlreadyPressesButton = false;
  }
  else {

    if (userAlreadyPressesButton == false) {
      // This is the situation where the user
      // has just now pressed down the button,
      // and was not doing it earlier.

      // tell to later loops that user is now
      // pressing the button:
      userAlreadyPressesButton = true;

      // Toggling:
      // if LED was not on, turn it on;
      // if LED was on, turn it off:
      if (ledIsOn == false) {
        digitalWrite(13,HIGH);
        ledIsOn = true;
      } else {
        digitalWrite(13,LOW);
        ledIsOn = false;
      }
    }
    else {
      // do nothing because this just means that
      // user has not lifted the finger from the
      // button yet.
    }
  }
}
```

```
void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    // if user is not pressing the button now,
    // tell that for later loops:
    userAlreadyPressesButton = false;
  }
  else {

    if (userAlreadyPressesButton == false) {
      // This is the situation where the user
      // has just now pressed down the button,
      // and was not doing it earlier.

      // tell to later loops that user is now
      // pressing the button:
      userAlreadyPressesButton = true;

      // Toggling:
      // if LED was not on, turn it on;
      // if LED was on, turn it off:
      if (ledIsOn == false) {
        digitalWrite(13,HIGH);
        ledIsOn = true;
      } else {
        digitalWrite(13,LOW);
        ledIsOn = false;
      }
    }
    else {
      // do nothing because this just means that
      // user has not lifted the finger from the
      // button yet.
    }
  }
}
```

It is very good idea to write comments that explain your code's logic

- Helps you remember your logic after a break
- Clarifies your thinking

This code has many if-else blocks that add intelligence to the program:

```
if (test is true) {
   commands
} else {
   commands
}
```

Blocks can "nest" inside each other

How to write good code

Writing your own functions that modularize the code 
Using commenting
Avoiding “hard coding”
Useful conventions

Writing your own functions that modularize the code

modularizing =

making the same code reusable from many places + making your code more understandable

Example:

This part takes many lines but does one elementary thing: it toggles the LED on and off.

This part would be more readable if we just wrote:

toggle(13);

```
void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    // if user is not pressing the button now,
    // tell that for later loops:
    userAlreadyPressesButton = false;
  }
  else {

    if (userAlreadyPressesButton == false) {
      // This is the situation where the user
      // has just now pressed down the button,
      // and was not doing it earlier.

      // tell to later loops that user is now
      // pressing the button:
      userAlreadyPressesButton = true;

      // Toggling:
      // if LED was not on, turn it on;
      // if LED was on, turn it off:
      if (ledIsOn == false) {
        digitalWrite(13,HIGH);
        ledIsOn = true;
      } else {
        digitalWrite(13,LOW);
        ledIsOn = false;
      }
    }
    else {
      // do nothing because this just means that
      // user has not lifted the finger from the
      // button yet.
    }
  }
}
```

Writing your own functions that modularize the code

```
void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    // if user is not pressing the button now,
    // tell that for later loops:
    userAlreadyPressesButton = false;
  }
  else {

    if (userAlreadyPressesButton == false) {
      // This is the situation where the user
      // has just now pressed down the button,
      // and was not doing it earlier.

      // tell to later loops that user is now
      // pressing the button:
      userAlreadyPressesButton = true;

      // Toggling:
      // if LED was not on, turn it on;
      // if LED was on, turn it off:
      toggle(13);
    }
    else {
      // do nothing because this just means that
      // user has not lifted the finger from the
      // button yet.
    }
  }
}
```

The code is now cleaner and easier to understand.

But how do we create this function so that we can use it this way?

toggle()

We announce in the program's beginning that we have our own function called toggle()

```
bool ledIsOn;
bool userAlreadyPressesButton;

void toggle(int pin);
```

Here is our code for setup() and loop()

```
// our setup() code
// (not copied here - see earlier slides)


// our loop() code
// (was shown in the previous slide)
```

We write the function that we announced anywhere in the code, such as to the end. In this case, we cut the code from loop() and pasted it here.

```
void toggle(int pin) {
  if (ledIsOn == false) {
    digitalWrite(pin,HIGH);
    ledIsOn = true;
  } else {
    digitalWrite(pin,LOW);
    ledIsOn = false;
  }
}
```

How to write good code

Writing own functions to modularize the code
Commenting ⇦
Avoiding “hard coding”
Useful conventions

Commenting

Can be used for two purposes:

To explain what the code does:

```
// detect if user is now pressing the button:
int buttonState = digitalRead(2);
```

When you test different effects, you can use comments to turn off and on some parts of code:

```
digitalWrite(13,LOW);
ledIsOn = 0;
// digitalWrite(13,HIGH);
// ledIsOn = 1;
```

There are two ways to make comments:

// Changes rest of the line into a comment

/* and */ change everything between these marks into a comment

```
/*
if (ledIsOn == false) {
  digitalWrite(pin,HIGH);
  ledIsOn = true;
} else {
  digitalWrite(pin,LOW);
  ledIsOn = false;
}
*/
```

How to write good code

Writing own functions to modularize the code
Commenting
Avoiding “hard coding” ⇦
Useful conventions

Avoiding “hard coding”

Consider this scenario:

- You have a complicated set of jump wires in your Arduino and breadboard
- You decide that you need to organize your wiring to make it more easily understandable
- You connect some wires to different pins in Arduino (such as, you move a wire from pin 2 to pin 8)

=> Problem emerges: You need to make lots of changes to your program code too

- Such as, where your code says pin 2, you have to change it to pin 8
- If you fail to make at least one of those changes, you have a bug in your code

Avoiding “hard coding”

Consider this scenario:

- You have a complicated set of jump wires in your Arduino and breadboard
- You decide that you need to organize your wiring to make it more easily understandable
- You connect some wires to different pins in Arduino (such as, you move a wire from pin 2 to pin 8)

=> Problem emerges: You need to make lots of changes to your program code too

- Such as, where your code says pin 2, you have to change it to pin 8
- If you fail to make at least one of those changes, you have a bug in your code

```
bool ledIsOn;
bool userAlreadyPressesButton;

void setup() {
  pinMode(2,INPUT);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
  ledIsOn = false;
  userAlreadyPressesButton = false;
}

void loop() {
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    userAlreadyPressesButton = false;
  }
  else {
    if (userAlreadyPressesButton == false) {
      userAlreadyPressesButton = true;
      toggle(13);
    }
    else {
      // do nothing
    }
  }
}

void toggle(int pin) {
  if (ledIsOn == false) {
    digitalWrite(pin,HIGH);
    ledIsOn = true;
  } else {
    digitalWrite(pin,LOW);
    ledIsOn = false;
  }
}
```

Avoiding “hard coding”…

Our LED toggler project has this problem:

- If you decide to rewire LED toggler’s wiring, you need to make lots of updates to your code
- Although your goal has been that rewiring makes your project more understandable, your program may stop working correctly

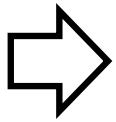
This problem results from “hard coding”

- = writing hard-to-change details into the code
- Hard coding can be avoided by using variables

```
bool ledIsOn;
bool userAlreadyPressesButton;

void setup() {
  pinMode(2,INPUT);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
  ledIsOn = false;
  userAlreadyPressesButton = false;
}

void loop() {
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    userAlreadyPressesButton = false;
  }
  else {
    if (userAlreadyPressesButton == false) {
      userAlreadyPressesButton = true;
      toggle(13);
    }
    else {
      // do nothing
    }
  }
}

void toggle(int pin) {
  if (ledIsOn == false) {
    digitalWrite(pin,HIGH);
    ledIsOn = true;
  } else {
    digitalWrite(pin,LOW);
    ledIsOn = false;
  }
}
```

How to remove and avoid hard coding

```
bool ledIsOn;
bool userAlreadyPressesButton;



void toggle(int pin);

void setup() {
  pinMode(2,INPUT);
  pinMode(13,OUTPUT);
  digitalWrite(13,LOW);
  ledIsOn = false;
  userAlreadyPressesButton = false;
}

void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(2);

  if (buttonState == LOW) {
    userAlreadyPressesButton = false;
  }
  else {
    if (userAlreadyPressesButton == false) {
      userAlreadyPressesButton = true;
      toggle(13);
    }
    else {
      // do nothing
    }
  }
}

void toggle(int pin) {
  if (ledIsOn == false) {
    digitalWrite(pin,HIGH);
    ledIsOn = true;
  } else {
    digitalWrite(pin,LOW);
    ledIsOn = false;
  }
}
```



```
bool ledIsOn;
bool userAlreadyPressesButton;

int buttonPin = 2;
int ledPin = 13;

void toggle(int pin);

void setup() {
  pinMode(buttonPin,INPUT);
  pinMode(ledPin,OUTPUT);
  digitalWrite(ledPin,LOW);
  ledIsOn = false;
  userAlreadyPressesButton = false;
}

void loop() {

  // detect if user is now pressing the button:
  int buttonState = digitalRead(buttonPin);

  if (buttonState == LOW) {
    userAlreadyPressesButton = false;
  }
  else {
    if (userAlreadyPressesButton == false) {
      userAlreadyPressesButton = true;
      toggle(ledPin);
    }
    else {
      // do nothing
    }
  }
}

void toggle(int pin) {
  if (ledIsOn == false) {
    digitalWrite(pin,HIGH);
    ledIsOn = true;
  } else {
    digitalWrite(pin,LOW);
    ledIsOn = false;
  }
}
```

When you rewire your jumpwires, you only need to change these two lines.

How to write good code

Writing own functions to modularize the code
Commenting
Avoiding “hard coding”
Useful conventions ⇦

Improving readability for humans

Use indents to visualize nested blocks:

```
void loop() {
  int buttonState = digitalRead(buttonInputPin);
  if (buttonState == HIGH) {
    if (userAlreadyPressesButton == 1) {
    }
    else {
      userAlreadyPressesButton = 1;

      if (ledIsOn == 0) {
        digitalWrite(ledPin,HIGH);
        ledIsOn = 1;
      } else {
        digitalWrite(ledPin,LOW);
        ledIsOn = 0;
      }
    }
  }
  else {
    userAlreadyPressesButton = 0;
  }
}
```

Indents used ☺

vs

```
void loop() {
int buttonState = digitalRead(buttonInputPin);
if (buttonState == HIGH) {
if (userAlreadyPressesButton == 1) {
}
else {
userAlreadyPressesButton = 1;

if (ledIsOn == 0) {
digitalWrite(ledPin,HIGH);
ledIsOn = 1;
} else {
digitalWrite(ledPin,LOW);
ledIsOn = 0;
}
}
}
else {
userAlreadyPressesButton = 0;
}
}
```

Indents not used ☹

Improving readability for humans

Use “camel case” in variable and function names:

- thisIsCamelCase

Alternative is “snake case”:

- this_is_snake_case

Use small letters in the variable and function names’ beginnings

- **N**otLikeThis
- Reason: the convention is that names starting with Big Letters are class names in object-oriented programming

Write your programs in English

- Write comments in English
- Use English variable and function names

Use variable and function names that describe their purpose

```
if (ledIsOn == false) {
  digitalWrite(ledPin,HIGH);
  ledIsOn = true;
} else {
  digitalWrite(ledPin,LOW);
  ledIsOn = false;
}
```

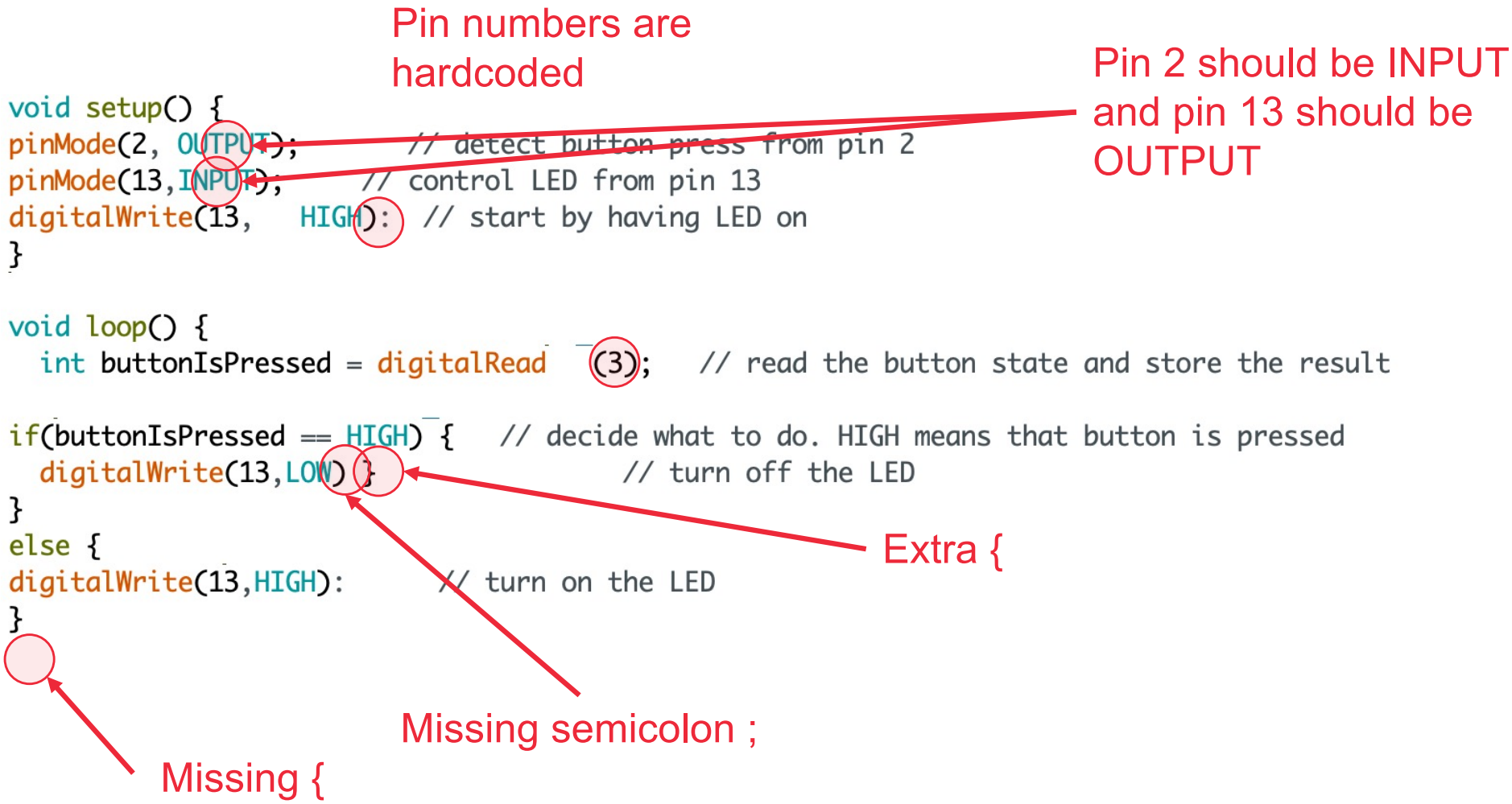
good

```
if (a == false) {
  digitalWrite(b,HIGH);
  c = true;
} else {
  digitalWrite(b,LOW);
  c = false;
}
```

bad

Exercise: find 8 errors

Here errors are harder to find because the code is not nicely indented. This is why correct indenting (i.e., use of tab characters) is important.



find-errors-exercise3.ino

What will we have on Monday 4 Oct?

Using a timer instead of a delay()

Reading input from sensors

- Pressure input

How to trouble-shoot bugs in our code