Capacitive Sensing PSET solutions

Computing Fabrics

Nov 8, 2022

Prof. Yoel Fink
Henry Cheung [Fibers@MIT]

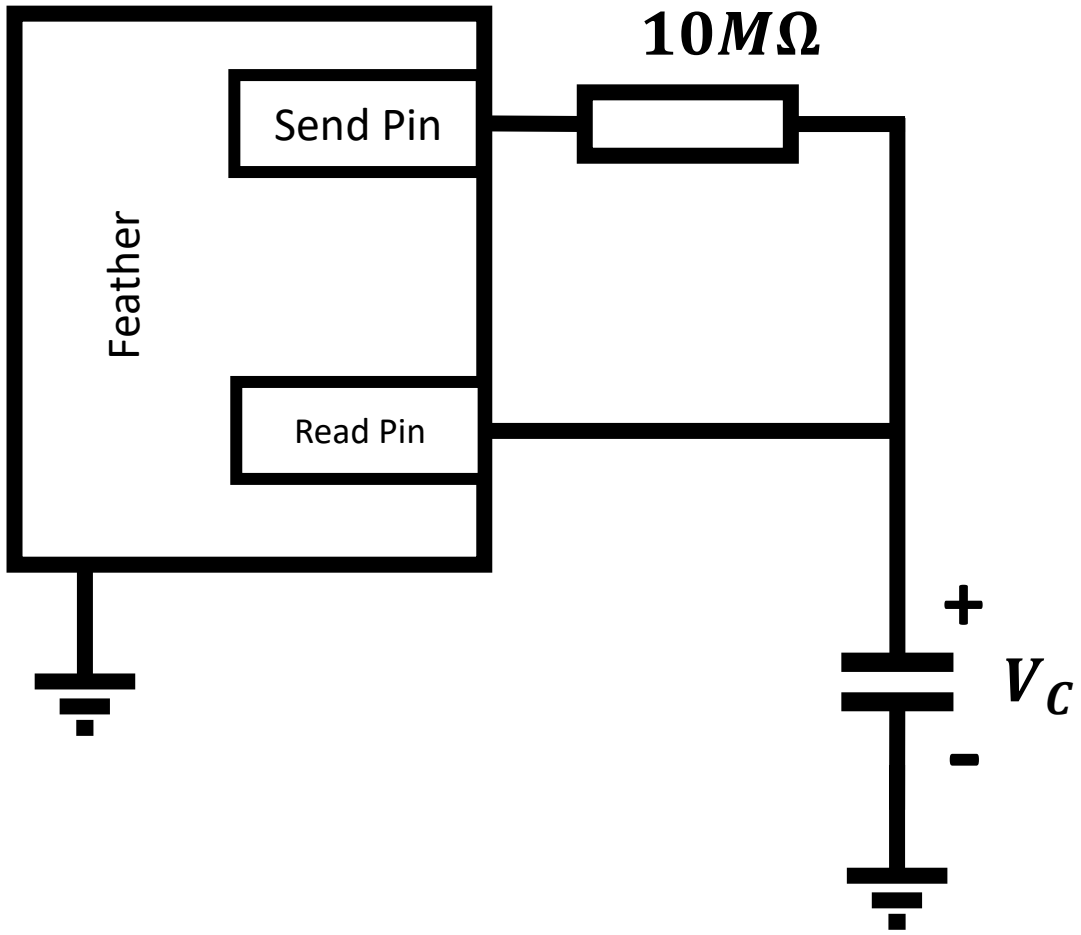
PSET questions

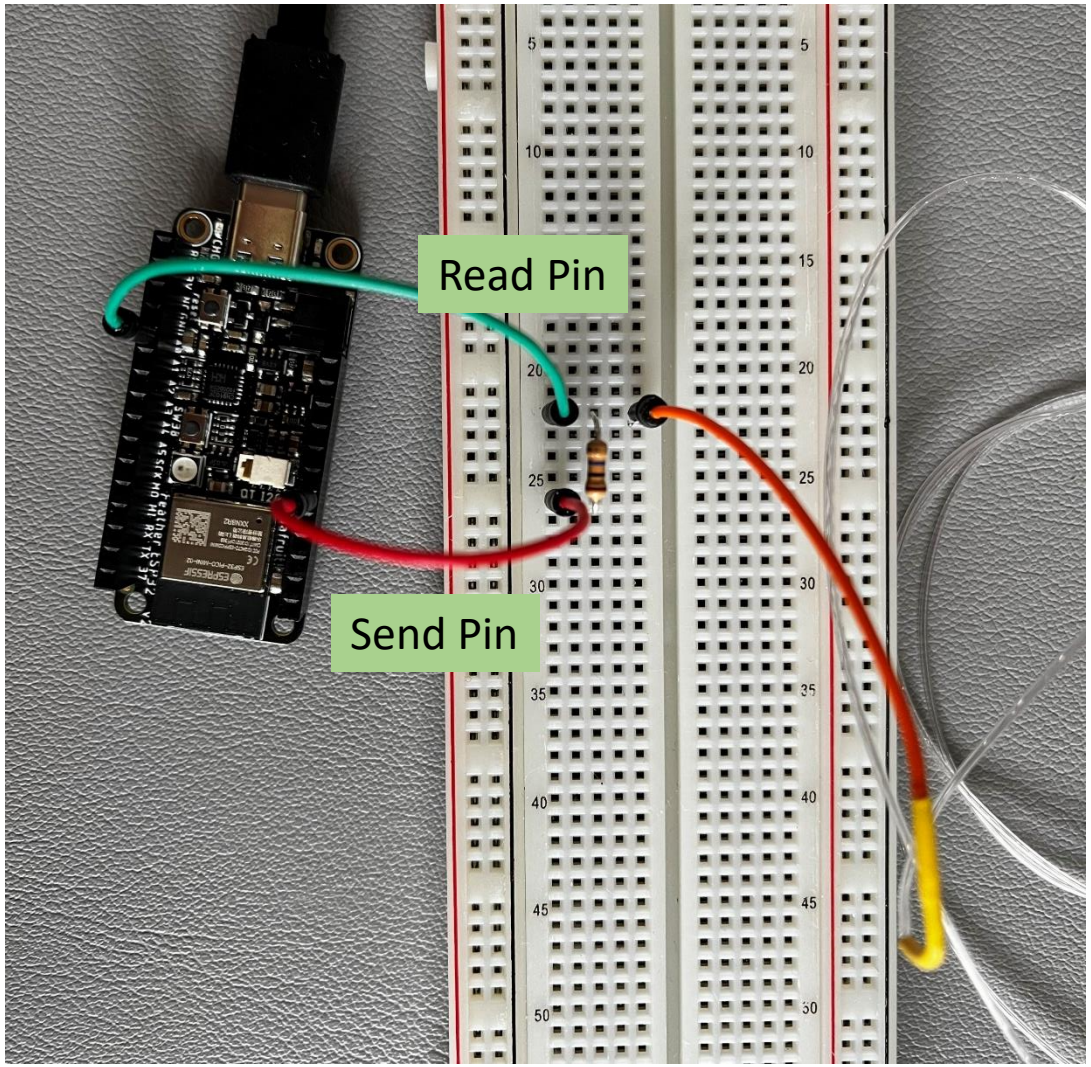
1. Detecting a hand and signaling
2. Filtering out high frequency noise
3. Determining the threshold between touch & hover
4. Detecting multiple taps
5. Checking left or right swipe over multiple sensors

Common pitfalls

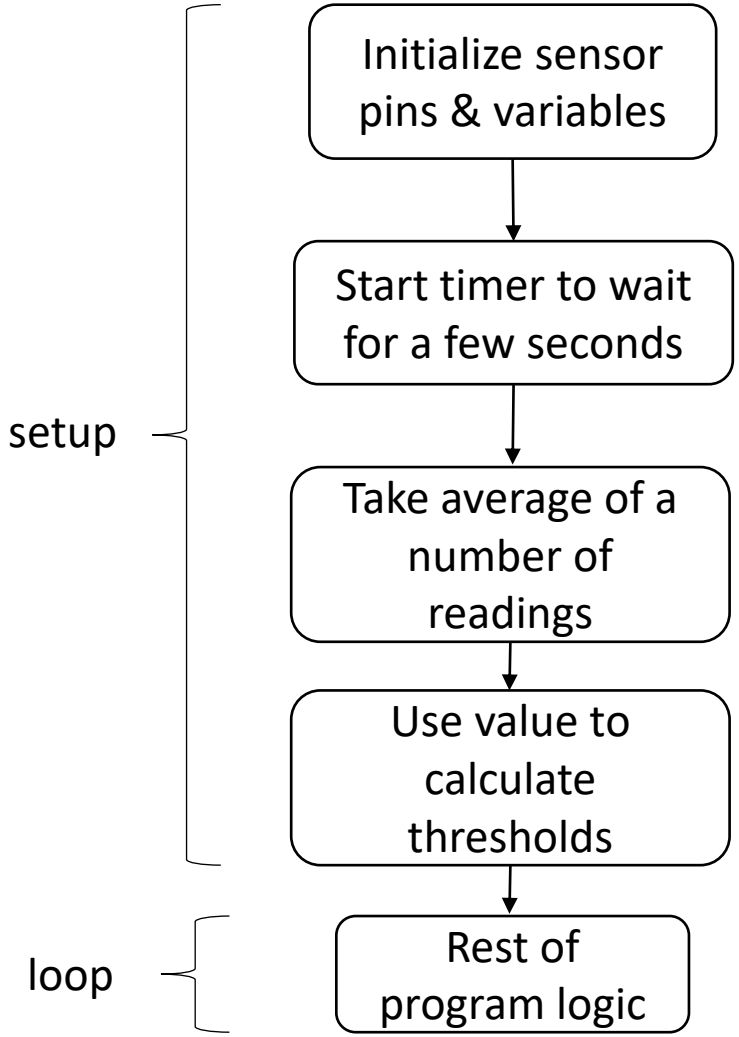
- Circuits were hooked up incorrectly
  - Wrong GPIO pins
  - Wrong wire to “capacitor” being sensed
- Capacitive sensors may need to be re-calibrated in different environments
- Selecting resistor values that are too low for the given capacitances
  - Charge time is too slow or too fast to see appreciable signals
- Noisy signal output needing longer averaging periods to filter out spurious readings

Circuit for 2 pin capacitive sensing

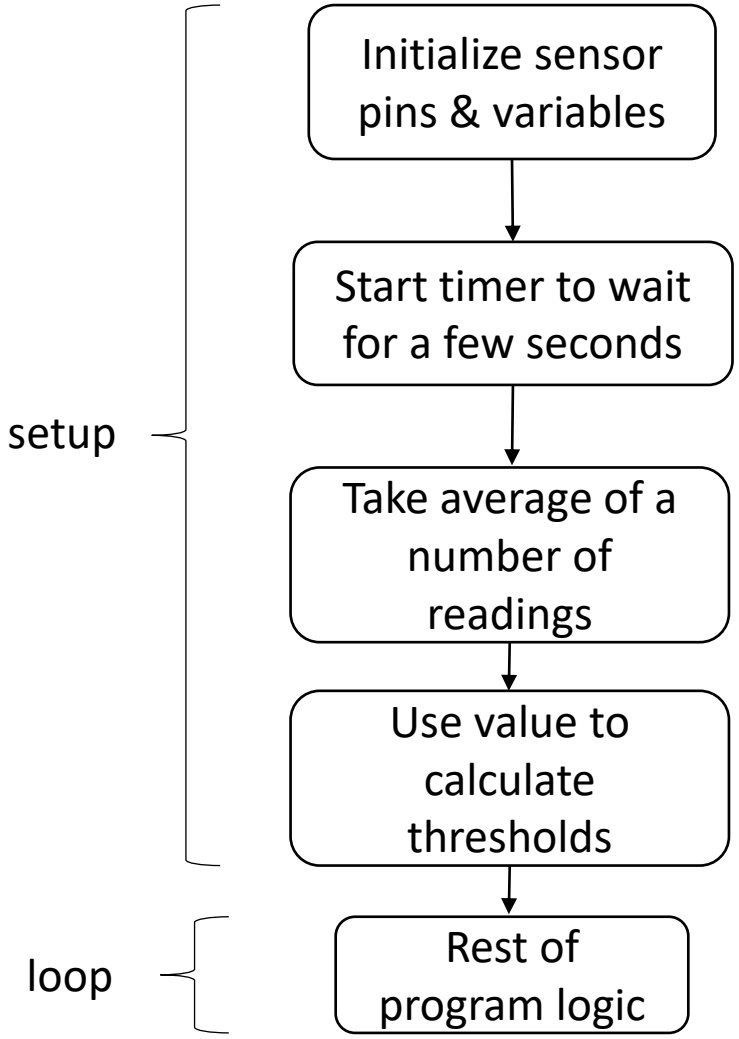




Calibration Setup

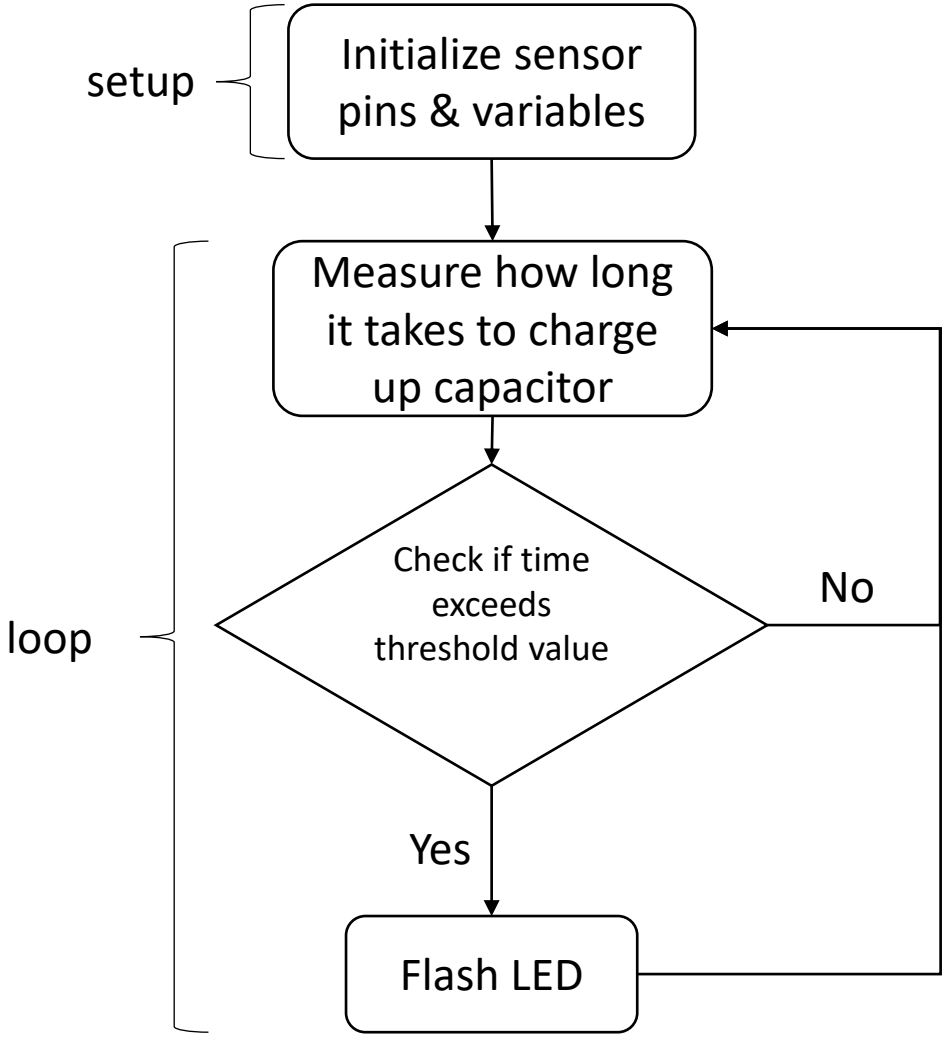


- Have a setup that averages a certain number of readings
- Use the averaged readings as your minimum threshold
- Calculate touch/hover thresholds as a certain percentage above them



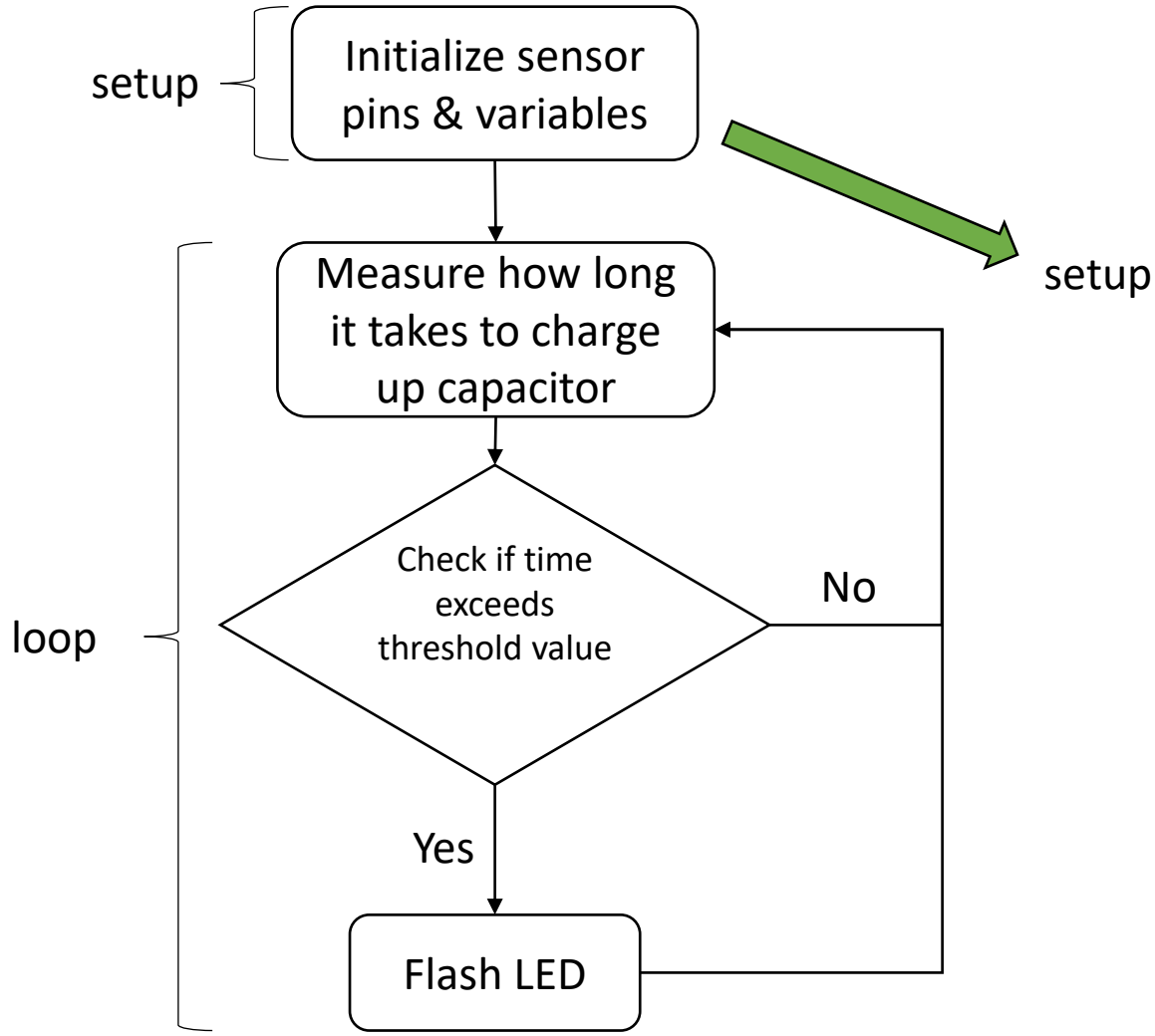

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 2, touchFactor = 5;

/*****************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  currentTime = millis();
  while (millis() - currentTime < waitTime) {
    Serial.println("WAIT");
    Serial.println(sensor.capacitiveSensor(numberOfSamples));
  }

  for (int counter = 0; counter < 10; counter++) {
    minThreshold += sensor.capacitiveSensor(numberOfSamples);
  }
  minThreshold /= 10;
  Serial.print("MIN THRESHOLD: ");
  Serial.println(minThreshold);

  hoverThreshold = hoverFactor * minThreshold;
  touchThreshold = touchFactor * minThreshold;
}

void loop()
{
  // PROGRAM LOGIC GOES HERE
}
```

Problem 1: Detect a hand nearby and signal



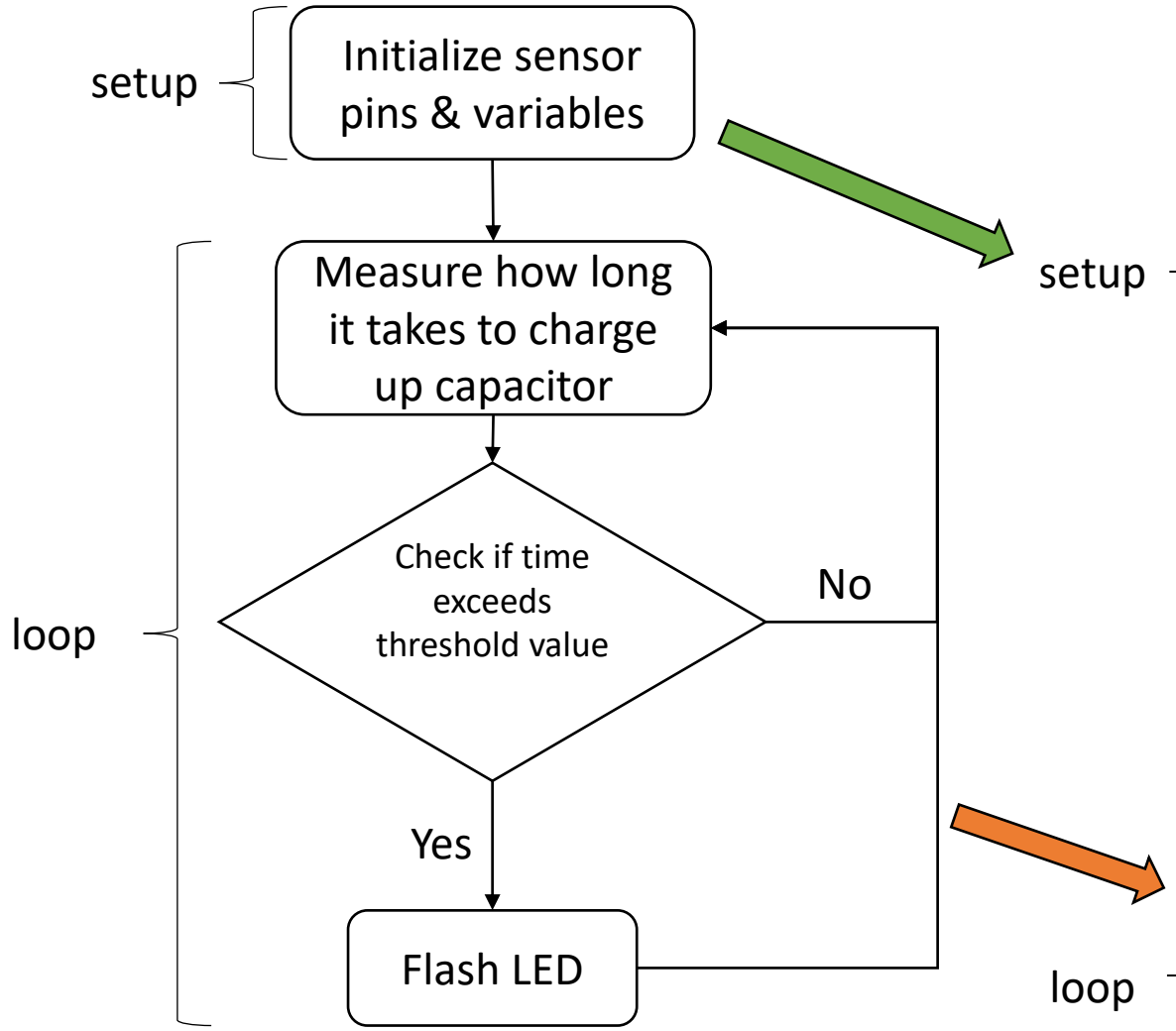
Setup:

- Checking on the capacitive fiber whether a certain threshold value has been met
- Signal on LED or serial port if condition met, else continue checking



setup

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 1.5, touchFactor = 2;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;

/*********************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```

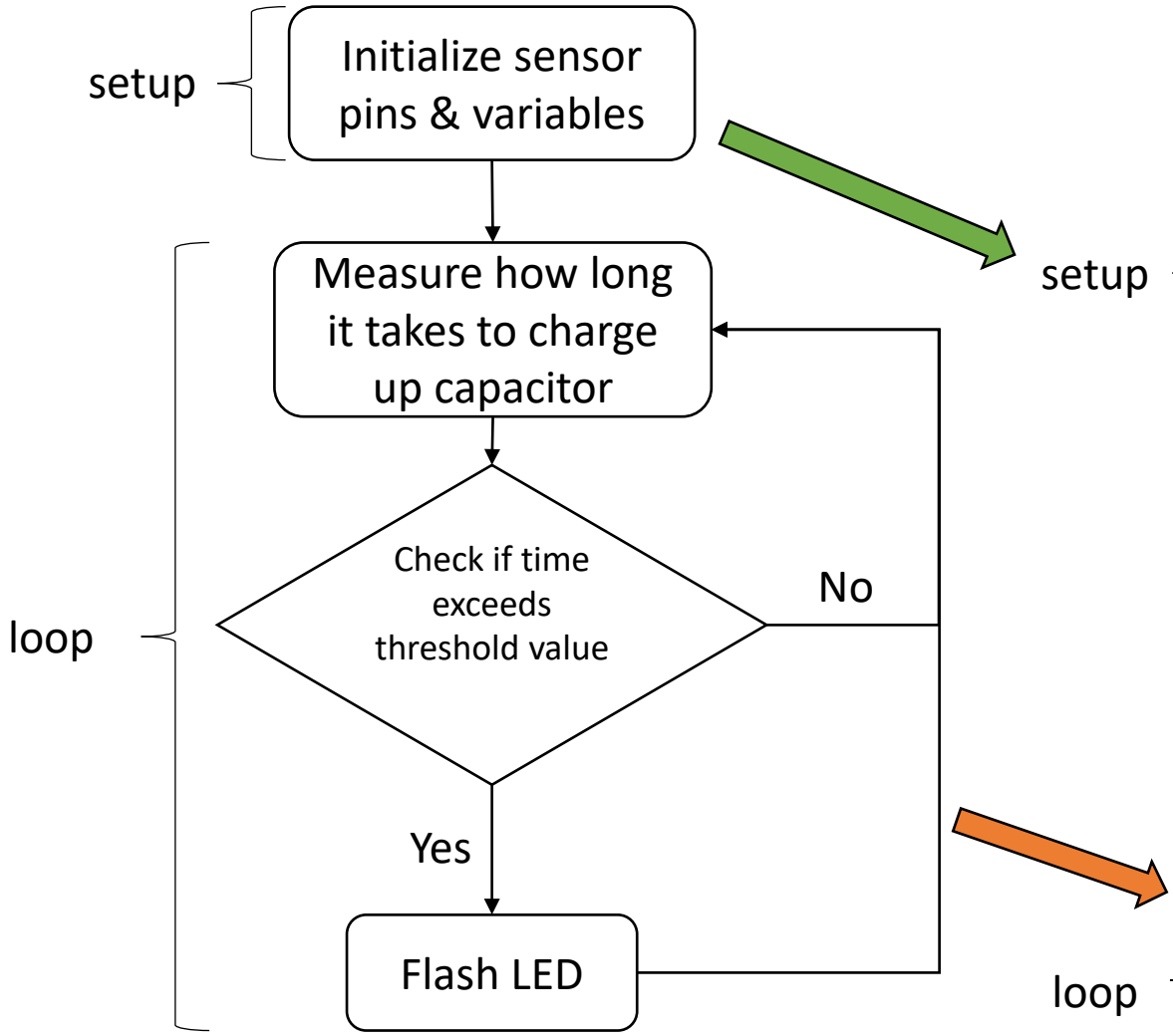


setup

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 1.5, touchFactor = 2;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;

/*********************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```

loop

```ino
void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  if (chargeTime > touchThreshold) {
    signalToUser();
  }

  Serial.println(chargeTime);
}
```



setup

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 1.5, touchFactor = 2;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;

/*********************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```

loop

```ino
void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  if (chargeTime > touchThreshold) {
    signalToUser();
  }

  Serial.println(chargeTime);
}
```

Functions in Arduino

- Functions are modular pieces of code that perform a defined task and then return to the area of code from which the function was “called”
- Typically used when the same action must be performed multiple times
- Advantages to using functions:
  - Helps conceptualize program
  - Actions that need to be repeated are only written once which reduces chances of making mistakes and only needs to be debugged once
  - Makes sketches smaller and more compact if sections of code are reused
  - Can allow code to be reused in other programs, just as we have been using functions from libraries like the CapacitiveSensor library

Function syntax in Arduino

- To create a function:
  1. Define the function name
  2. Create names for variables to be passed in
     - Functions do not require inputs/outputs, they can simply execute a piece of code
  3. Define the data type that will be returned, if any. If there is nothing to be returned, use *"void"*
     - *"void"* should be familiar as the 2 functions that every Arduino sketch contains are *"setup()"* and *"loop()"* as the microcontroller essentially calls those functions to execute when turned on

```
datatype functionName(parameter list) {
        lines of code
        optional return statement;
}

int myMultiplyFunction(int x, int y) {
        int result;
        result = x * y;
        return result;
}
```

Function syntax in Arduino

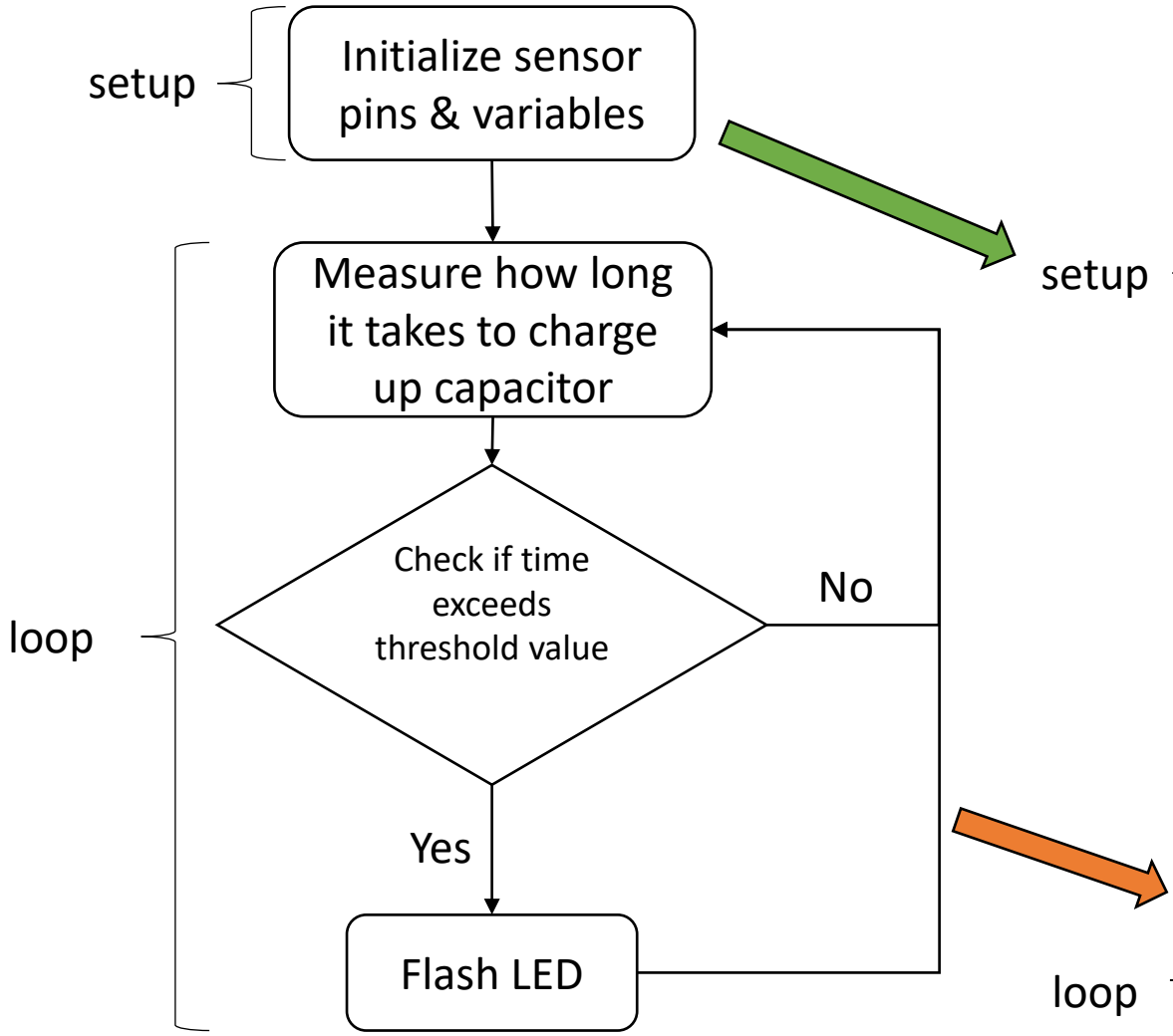
- To call a function:
  1. Make sure your function has been defined at the bottom of your sketch
  2. Type the "functionName(optional input variables)"
  3. Don't forget to assign the return result to a variable if you need to use it or pass the value to another function

```
datatype functionName(parameter list) {
        lines of code
        optional return statement;
}

int myMultiplyFunction(int x, int y) {
        int result;
        result = x * y;
        return result;
}
```

Function syntax in Arduino

- To call a function:
  1. Make sure your function has been defined at the bottom of your sketch
  2. Type the "functionName(optional input variables)"
  3. Don't forget to assign the return result to a variable if you need to use it or pass the value to another function

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

void loop() {
  int i = 2;
  int j = 3;
  int k;

  k = myMultiplyFunction(i, j); // k now contains 6
  Serial.println(k);
  delay(500);
}

int myMultiplyFunction(int x, int y){
  int result;
  result = x * y;
  return result;
}
```



setup

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 1.5, touchFactor = 2;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;

/*********************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```

loop

```ino
void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  if (chargeTime > touchThreshold) {
    signalToUser();
  }

  Serial.println(chargeTime);
}
```

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 1.5, touchFactor = 2;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;

/********************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}

void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  if (chargeTime > touchThreshold) {
    signalToUser();
  }

  Serial.println(chargeTime);
}
```

Function definitions at the bottom of the file

```ino
/************** HELPER FUNCTIONS ***************/

/**
 * Signals to user by flashing a LED fiber
 */
void signalToUser() {
  digitalWrite(LED, HIGH);
  delay(signalDelay);
  digitalWrite(LED, LOW);
}

/**
 * Initializes LED GPIO pin and sets the LED as off initially
 */
void setupLED() {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
}

/**
 * Throws away the first few seconds of data
 */
void setupThresholds() {
  currentTime = millis();
  while (millis() - currentTime < waitTime) {
    Serial.println("WAIT");
    Serial.println(sensor.capacitiveSensor(numberOfSamples));
  }

  for (int counter = 0; counter < 10; counter++) {
    minThreshold += sensor.capacitiveSensor(numberOfSamples);
  }
  minThreshold /= 10;
  Serial.print("MIN THRESHOLD: ");
  Serial.println(minThreshold);

  hoverThreshold = hoverFactor * minThreshold;
  touchThreshold = touchFactor * minThreshold;
}
```

Problem 1 review

Main Pointers:

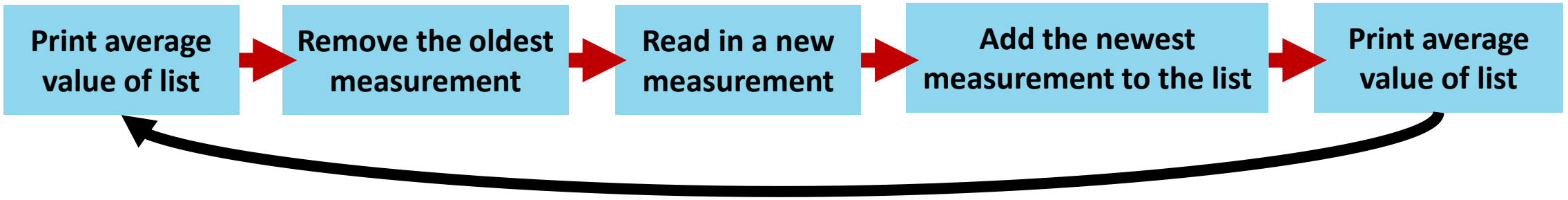
- Use functions to make code more readable
- NO “magic numbers”, use constant variables near the top of the file
- Always comment!
  - Like spice, use in moderation for best effect

```
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 1.5, touchFactor = 2;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;

/*****************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}

void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  if (chargeTime > touchThreshold) {
    signalToUser();
  }

  Serial.println(chargeTime);
}
```

Problem 2: Filtering out high frequency noise

- Solution:
  - Use the ESP_CapSense library's capacitiveSensor() function's built-in averaging
  - Implement the moving average filter
  - Use provided low pass filter code

Setup: Create array/list of size N



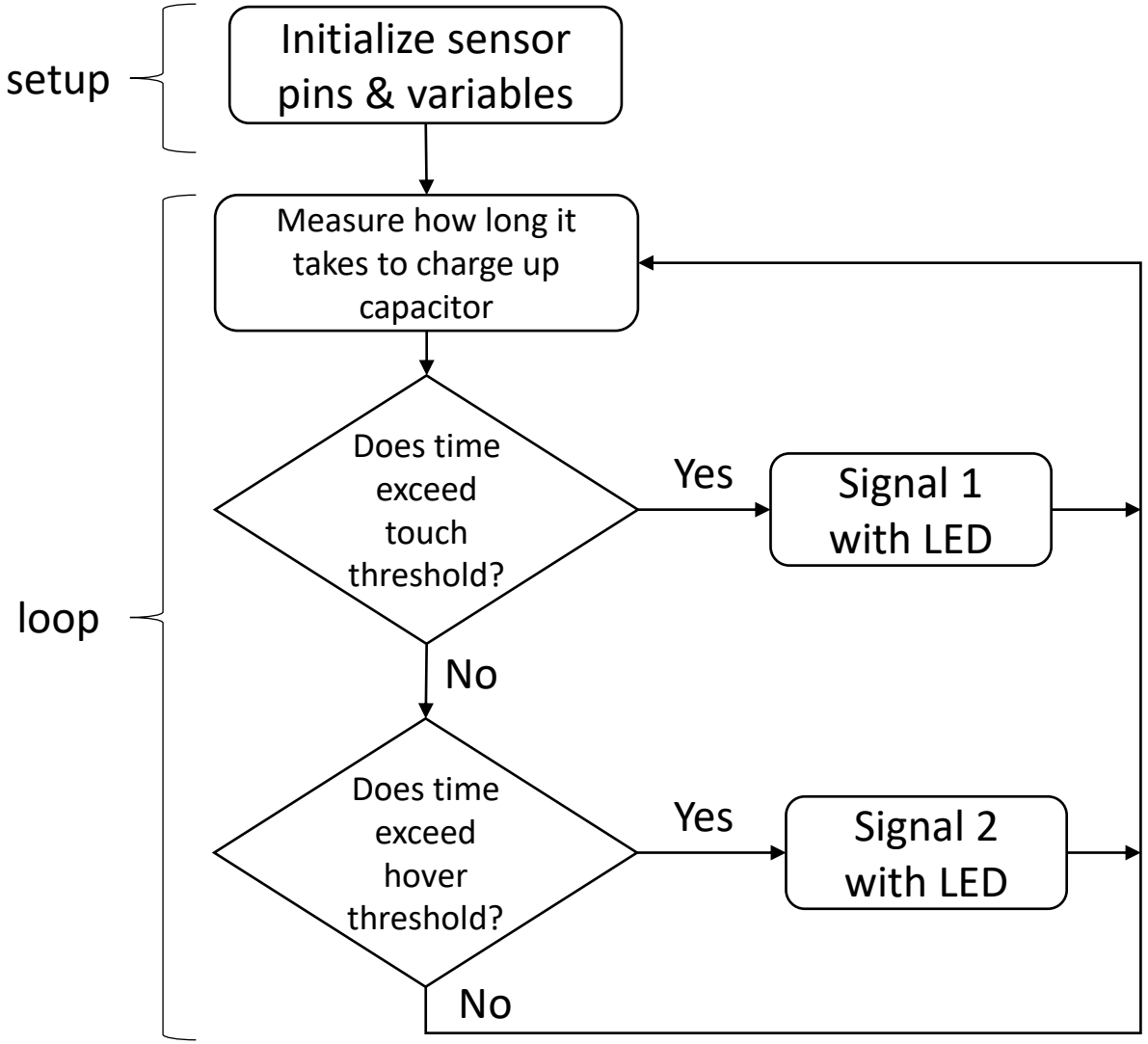
Problem 2 review

Main Pointers:

- Difference equation requires you to store past values of the filtered output and previous charge times in global variables
- “counter % 3 == 0” is simply a construct that lets us skip every 3 samples

```
#include <ESP_CapSense.h>

...

int counter = 0;

int prevChargeTime, prevOutput, prevChargeTime;
float FILTER_COEFF[] = [0.969, 0.0155, 0.0155];

/*****************************************/

void setup()
{
  ...
}

void loop()
{
  // total time to charge from function with set number of samples
  int currentChargeTime = sensor.capacitiveSensor(numberOfSamples);

  // Computing a weighted signal, biased to not shift very much from previous values
  float currentOutput = FILTER_COEFF[0] * prevOutput + FILTER_COEFF[1] *
currentChargeTime + FILTER_COEFF[2] * prevChargeTime;

  // Updating the previous values
  prevChargeTime = currentChargeTime;
  prevOutput = currentOutput;

  // Only uses every third data point as the previous 2 are used to calculate the
filter data point
  if (counter % 3 == 0) {
    if (currentOutput > touchThreshold) {
      signalToUser(LONGER);
    } else if (currentOutput > hoverThreshold) {
      signalToUser(SHORTER);
    }

    Serial.println(currentOutput);
  }
  counter++;
}
```

Problem 3: Multiple Thresholds



Setup:

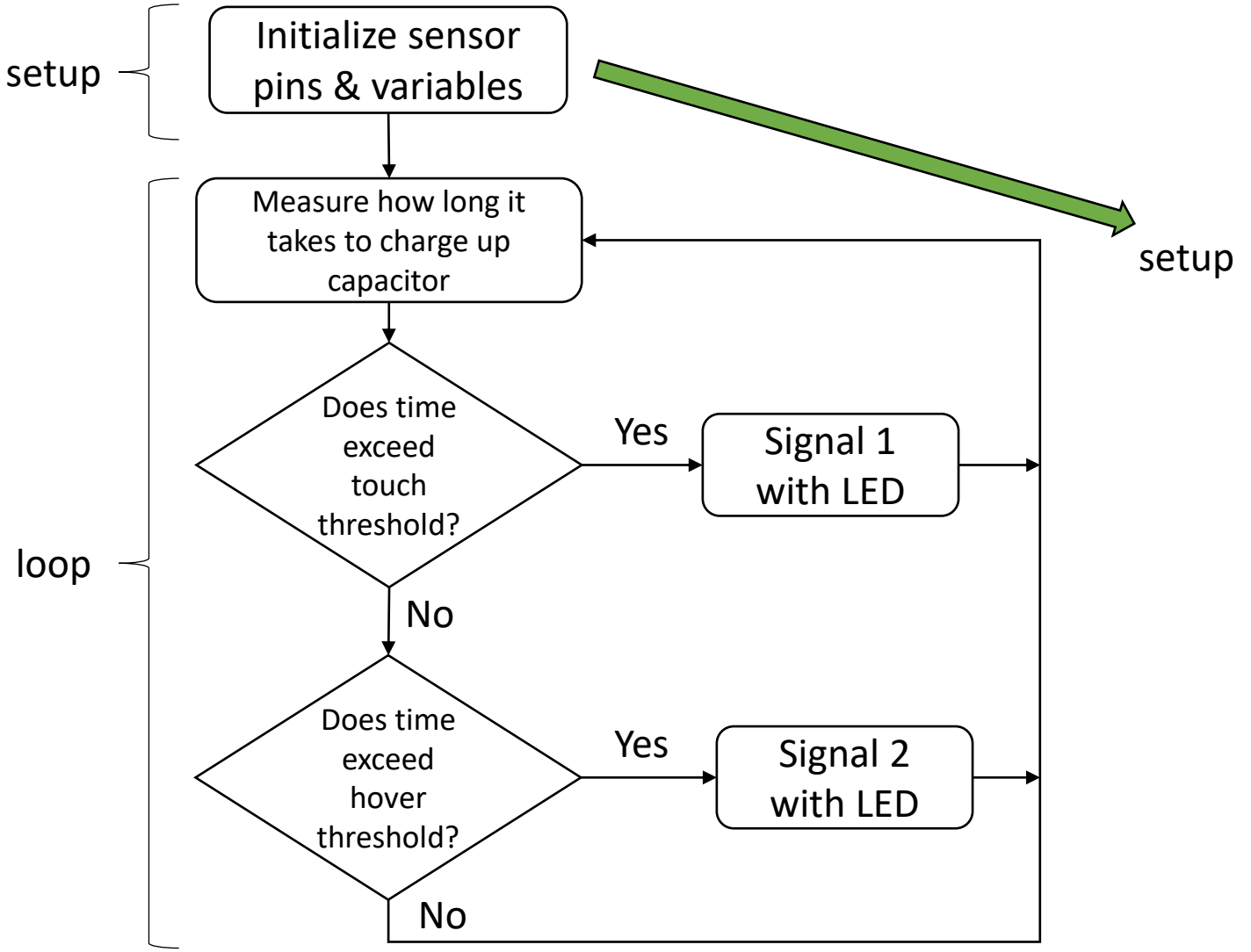
- Extend your program from part (1) to distinguish between a hand touch versus a hand hovering. This will require calibrating the system based on observed values and possibly other techniques, like the averaging function that Juliana demonstrated in class.

Problem 3: Multiple Thresholds

Setup:

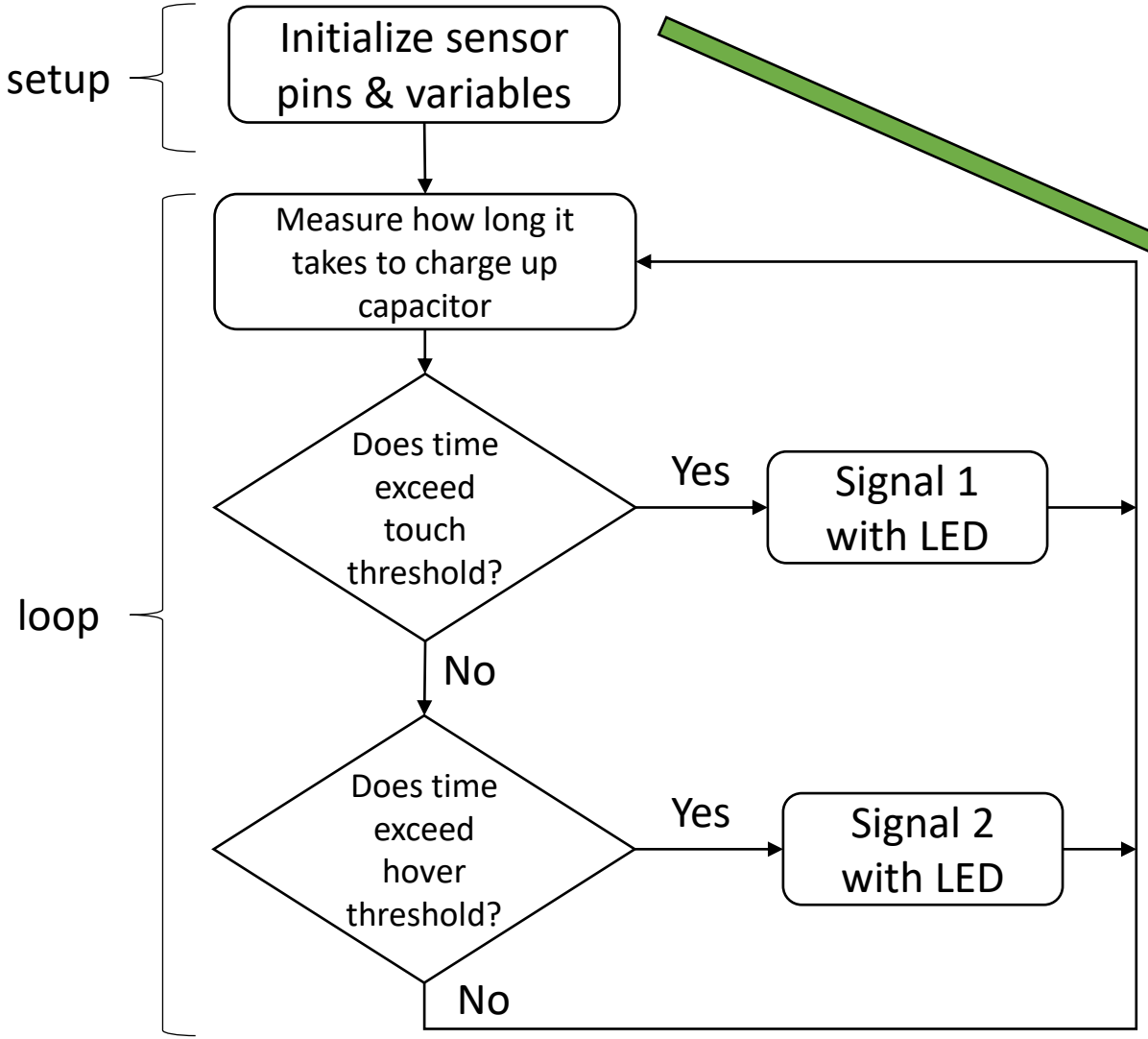
- Extend your program from part (1) to distinguish between a hand touch versus a hand hovering. This will require calibrating the system based on observed values and possibly other techniques, like the averaging function that Juliana demonstrated in class.

Hardware: same as before!



setup

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 2, touchFactor = 5;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;
const int LONGER = 1, LONGER_FACTOR = 5;
const int SHORTER = 2;

/*********************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```



setup

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 2, touchFactor = 5;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;
const int LONGER = 1, LONGER_FACTOR = 5;
const int SHORTER = 2;

/*****************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```

loop

```ino
void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  if (chargeTime > touchThreshold) {
    signalToUser(SHORTER);
  } else if (chargeTime > hoverThreshold) {
    signalToUser(LONGER);
  }

  Serial.println(chargeTime);
}
```

```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime;
int waitTime = 10000; // 10 seconds

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 2, touchFactor = 5;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;
const int LONGER = 1, LONGER_FACTOR = 5;
const int SHORTER = 2;

/********************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}

void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  if (chargeTime > touchThreshold) {
    signalToUser(SHORTER);
  } else if (chargeTime > hoverThreshold) {
    signalToUser(LONGER);
  }

  Serial.println(chargeTime);
}
```

Function definitions at the bottom of the file

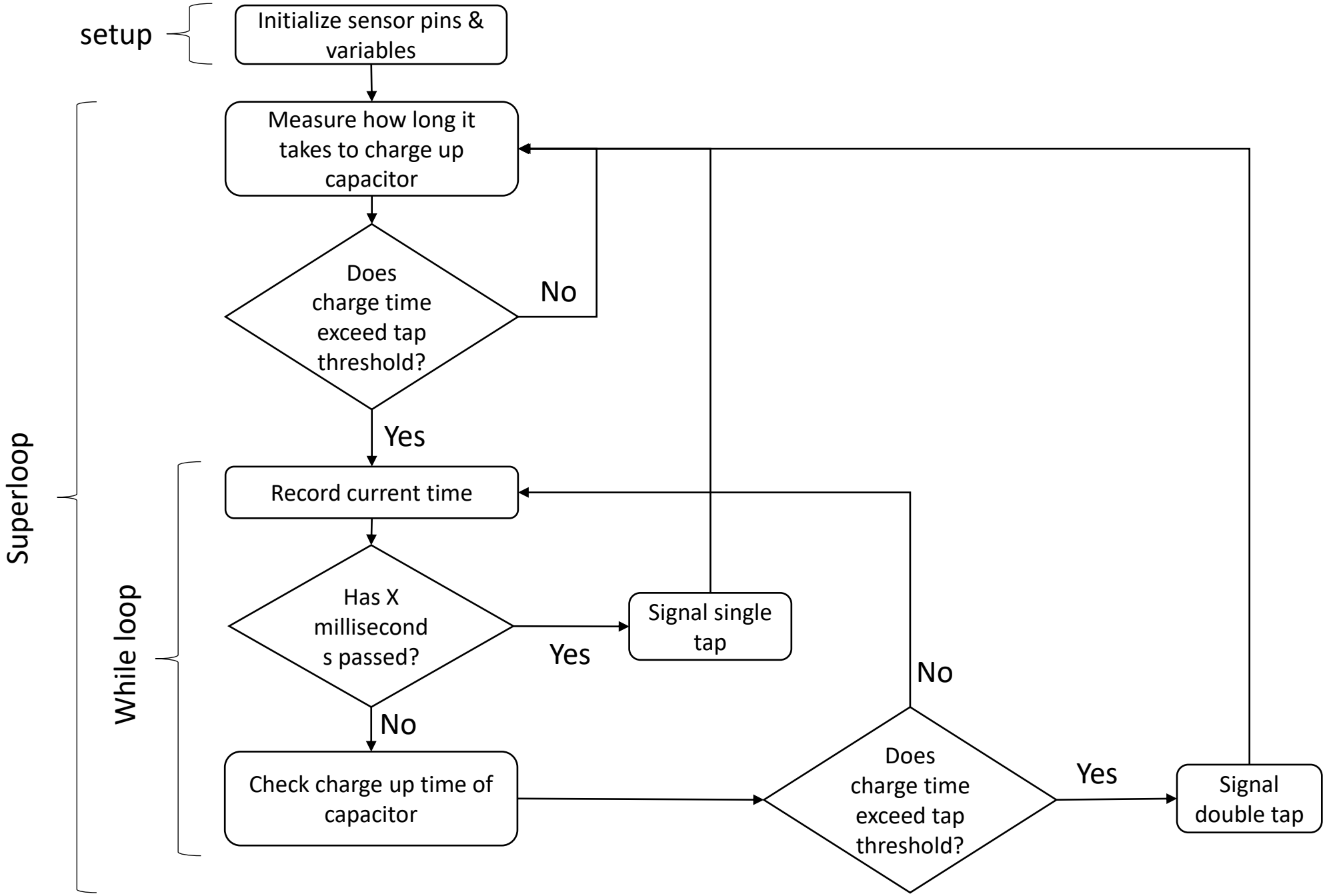
```ino
/*************** HELPER FUNCTIONS ***************/

/**
 * Signals to user by flashing a LED fiber
 */
void signalToUser(int type) {
  if (type == LONGER) {
      digitalWrite(LED, HIGH);
      delay(LONGER_FACTOR * signalDelay);
      digitalWrite(LED, LOW);
      delay(LONGER_FACTOR * signalDelay);
  } else if (type == SHORTER) {
      digitalWrite(LED, HIGH);
      delay(signalDelay);
      digitalWrite(LED, LOW);
  }
}

/**
 * Initializes LED GPIO pin and sets the LED as off initially
 */
void setupLED() {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
}

/**
 * Throws away the first few seconds of data
 * User should NOT touch the fiber upon startup
 * Then, calculates the hover/touch thresholds by a certain percentage above the noise floor
 */
void setupThresholds() {
  currentTime = millis();
  while (millis() - currentTime < waitTime) {
    Serial.println("WAIT");
    Serial.println(sensor.capacitiveSensor(numberOfSamples));
  }

  for (int counter = 0; counter < 10; counter++) {
    minThreshold += sensor.capacitiveSensor(numberOfSamples);
  }
  minThreshold /= 10;
  Serial.print("MIN THRESHOLD: ");
  Serial.println(minThreshold);

  hoverThreshold = hoverFactor * minThreshold;
  touchThreshold = touchFactor * minThreshold;
}
```

Problem 4: classifying different inputs

Setup:

- Extend your program from part (2) to detect at least 2 or 3 different types of input (e.g. number of taps from a hand, etc.) and uniquely signal that each has been recognized.

Hardware: same as before!


setup
Initialize sensor pins & variables
Superloop
Measure how long it takes to charge up capacitor
Does charge time exceed tap threshold?
No
Yes
While loop
Record current time
Has X millisecond s passed?
Yes
Signal single tap
No
Check charge up time of capacitor
Does charge time exceed tap threshold?
No
Yes
Signal double tap

Other loops in Arduino

- While loops

Syntax

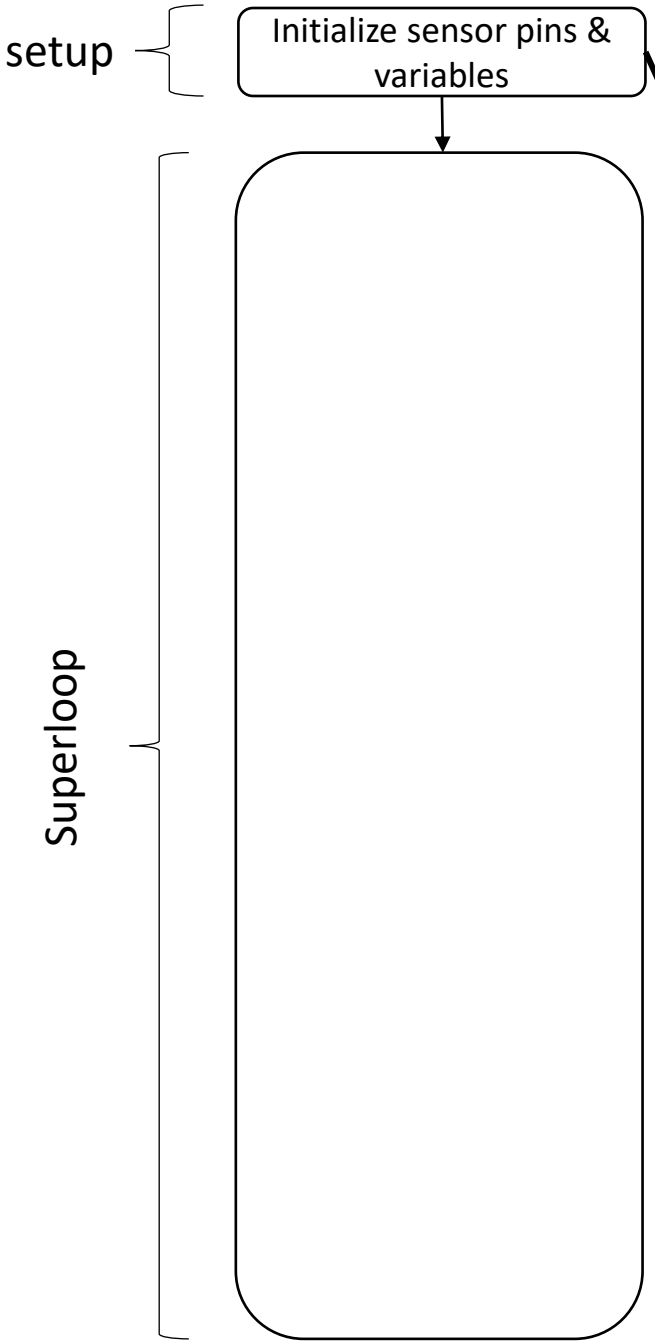
```
while (condition) {
  // statement(s)
}
```

Parameters

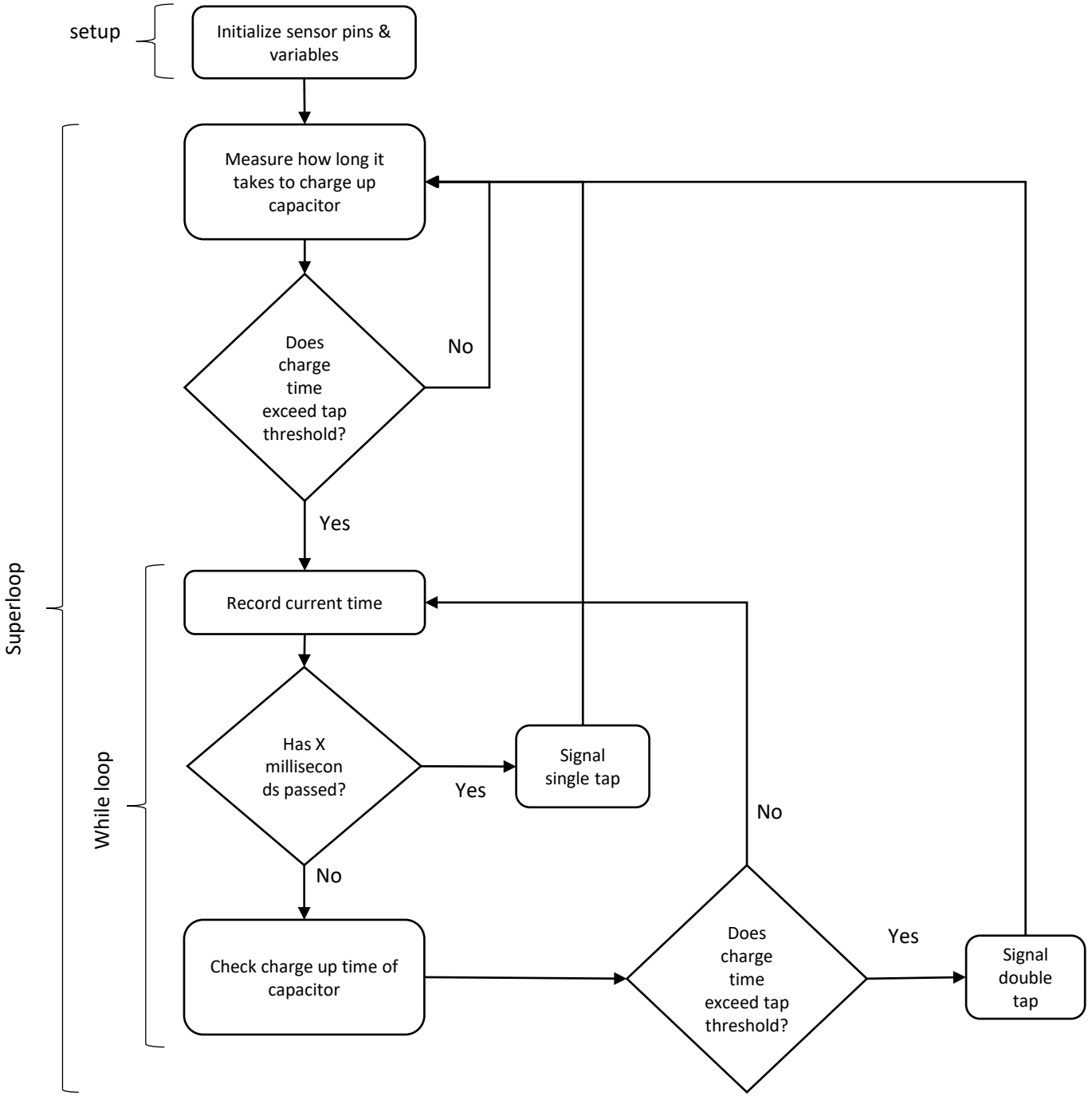
`condition`: a boolean expression that evaluates to `true` or `false`.

Example Code

```
var = 0;
while (var < 200) {
  // do something repetitive 200 times
  var++;
}
```




```ino
#include <ESP_CapSense.h>

const int sendPin = 4, receivePin = 33, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor sensor = CapacitiveSensor(sendPin,receivePin);

// timer variable [milliseconds]
unsigned long currentTime, startTime;
int waitTime = 5000; // 5 seconds

const int doubleTapWaitTime = 500;
const int tapDebounceTime = 100;

// threshold variables
int minThreshold = 0, hoverThreshold = 0, touchThreshold = 0;
float hoverFactor = 2, touchFactor = 5;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;
bool doubleTapFlag = false;

/*****************************************/

void setup()
{
  sensor.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```




```
void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  // A tap was detected
  if (chargeTime > touchThreshold) {
    // Wait to allow user to withdraw their hand for another tap
    delay(tapDebounceTime);

    // start timer
    startTime = millis();

    while( (millis() - startTime) < doubleTapWaitTime) {
      chargeTime = sensor.capacitiveSensor(numberOfSamples);
      if (chargeTime > touchThreshold) {
        doubleTapFlag = true;
        break;
      }
    }

    if (doubleTapFlag) {
      signalToUser();
      signalToUser();
      signalToUser();
      signalToUser();
      signalToUser();
      // resetting flag
      doubleTapFlag = false;
    }
  }

  Serial.println(chargeTime);
}
```

```
void loop()
{
  // total time to charge from function with set number of samples
  int chargeTime = sensor.capacitiveSensor(numberOfSamples);

  // A tap was detected
  if (chargeTime > touchThreshold) {
    // Wait to allow user to withdraw their hand for another tap
    delay(tapDebounceTime);

    // start timer
    startTime = millis();

    while( (millis() - startTime) < doubleTapWaitTime) {
      chargeTime = sensor.capacitiveSensor(numberOfSamples);
      if (chargeTime > touchThreshold) {
        doubleTapFlag = true;
        break;
      }
    }

    if (doubleTapFlag) {
      signalToUser();
      signalToUser();
      signalToUser();
      signalToUser();
      signalToUser();
      // resetting flag
      doubleTapFlag = false;
    }
  }

  Serial.println(chargeTime);
}
```

```
/*************** HELPER FUNCTIONS ***************/

/**
 * Signals to user by flashing a LED fiber
 */
void signalToUser() {
  digitalWrite(LED, HIGH);
  delay(signalDelay);
  digitalWrite(LED, LOW);
}

/**
 * Initializes LED GPIO pin and sets the LED as off initially
 */
void setupLED() {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
}

/**
 * Throws away the first few seconds of data
 * User should NOT touch the fiber upon startup
 * Then, calculates the hover/touch thresholds by a certain percentage
above the noise floor
 */
void setupThresholds() {
  currentTime = millis();
  while (millis() - currentTime < waitTime) {
    Serial.println("WAIT");
    Serial.println(sensor.capacitiveSensor(numberOfSamples));
  }

  for (int counter = 0; counter < 10; counter++) {
    minThreshold += sensor.capacitiveSensor(numberOfSamples);
  }
  minThreshold /= 10;
  Serial.print("MIN THRESHOLD: ");
  Serial.println(minThreshold);

  hoverThreshold = hoverFactor * minThreshold;
  touchThreshold = touchFactor * minThreshold;
}
```

Other loops in Arduino

- For loops

Syntax

```
for (initialization; condition; increment) {
  // statement(s);
}
```

Parameters

`initialization`: happens first and exactly once.

`condition`: each time through the loop, `condition` is tested; if it's `true`, the statement block, and the **increment** is executed, then the **condition** is tested again. When the **condition** becomes `false`, the loop ends.

`increment`: executed each time through the loop when `condition` is `true`.

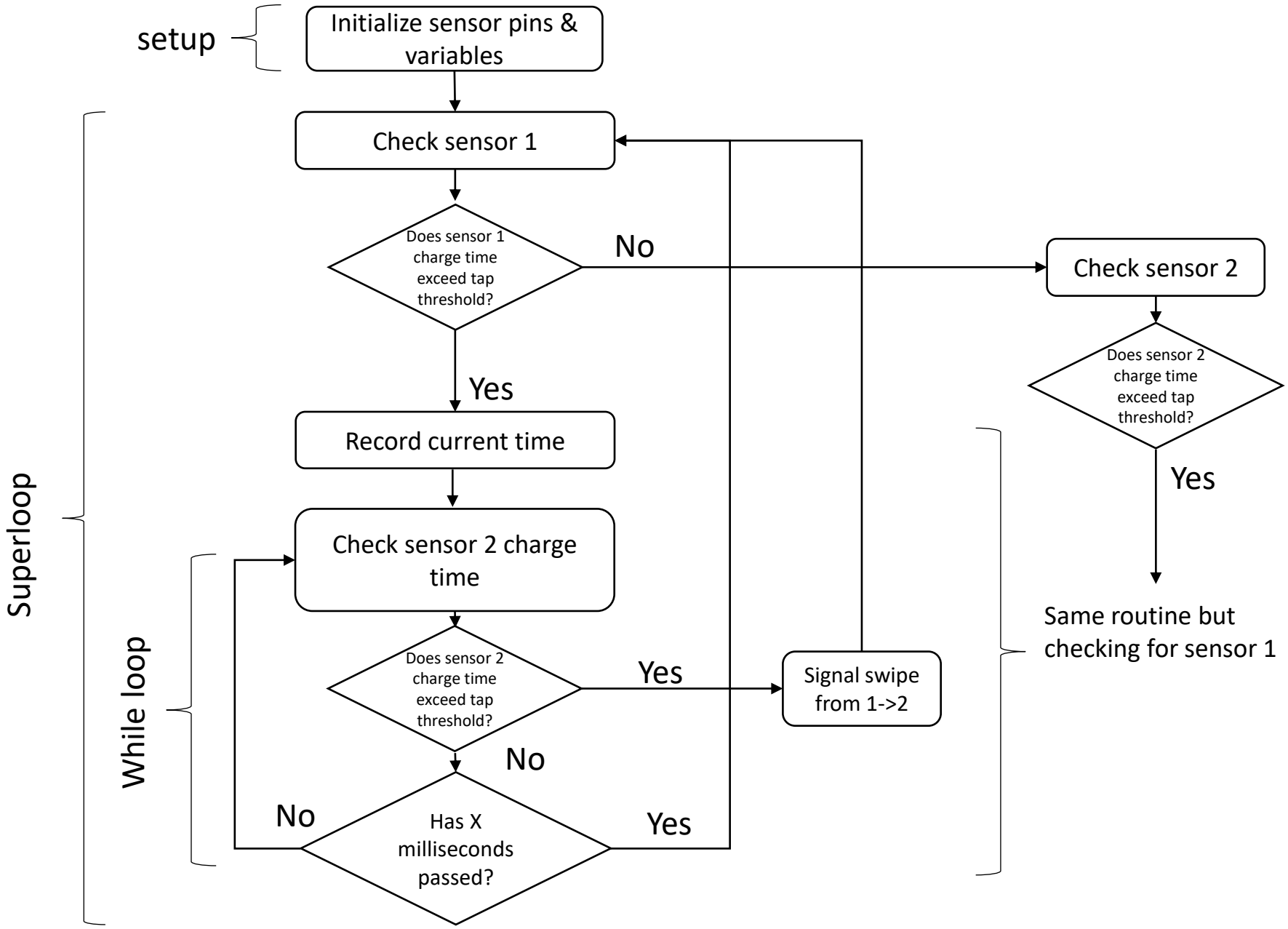
Example Code

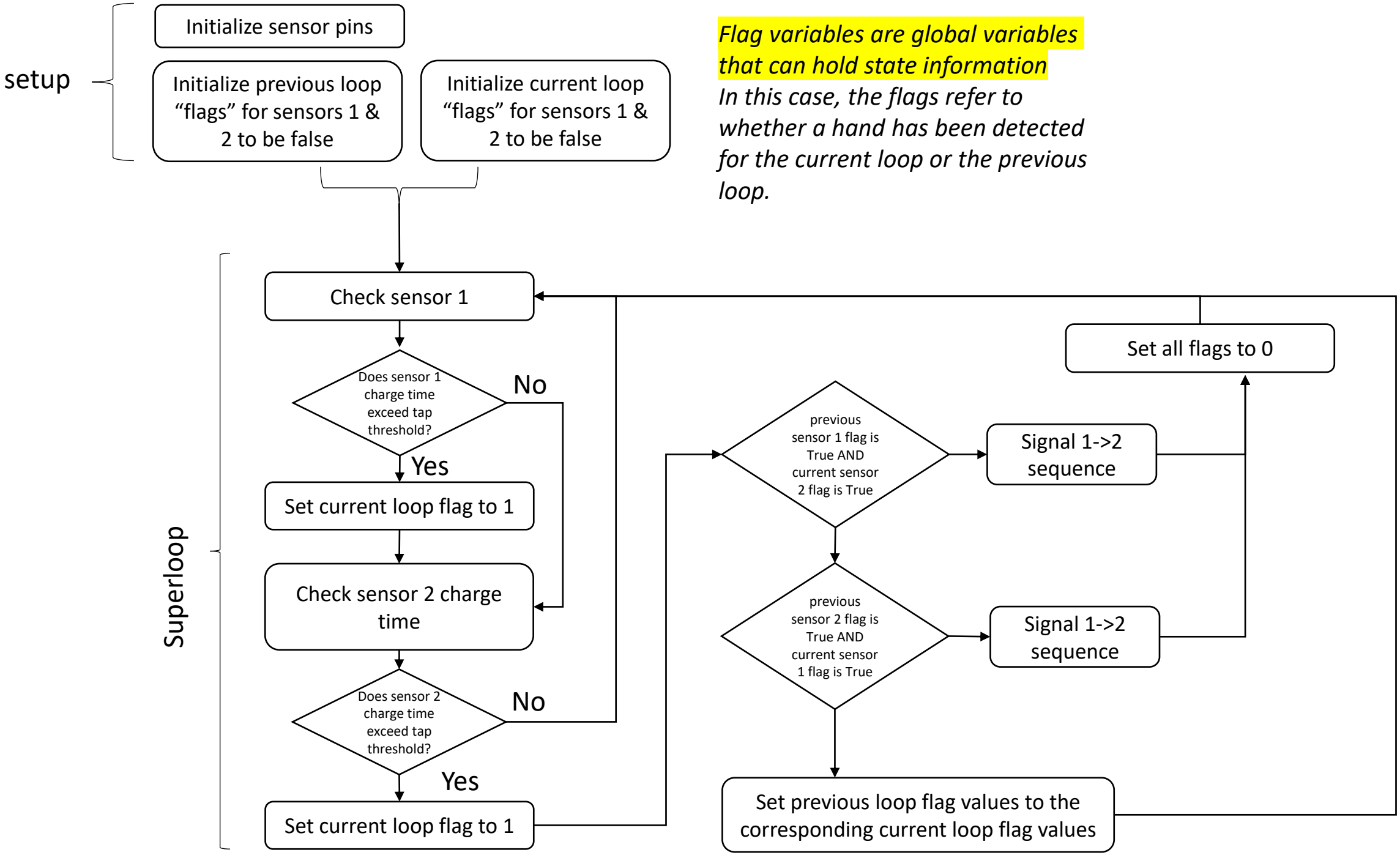
```
// Dim an LED using a PWM pin
int PWMpin = 10;  // LED in series with 470 ohm resistor on pin 10

void setup() {
  // no setup needed
}

void loop() {
  for (int i = 0; i <= 255; i++) {
    analogWrite(PWMpin, i);
    delay(10);
  }
}
```

Problem 5: Gestures

Setup:

- Read input from multiple sensors
- When a sensor detects a hand nearby, check if a sequence of other sensors detect a hand nearby within a certain time period
- Depending on the sequence of sensors that are activated, signal to the user either a left swipe, right swipe, etc.


setup
Initialize sensor pins & variables
Superloop
Check sensor 1
Does sensor 1 charge time exceed tap threshold?
No
Yes
Record current time
While loop
Check sensor 2 charge time
Does sensor 2 charge time exceed tap threshold?
Yes
No
Has X milliseconds passed?
No
Yes
Signal swipe from 1->2
Check sensor 2
Does sensor 2 charge time exceed tap threshold?
Yes
Same routine but checking for sensor 1


setup
Initialize sensor pins
Initialize previous loop "flags" for sensors 1 & 2 to be false
Initialize current loop "flags" for sensors 1 & 2 to be false
Flag variables are global variables that can hold state information
In this case, the flags refer to whether a hand has been detected for the current loop or the previous loop.
Superloop
Check sensor 1
Does sensor 1 charge time exceed tap threshold?
No
Yes
Set current loop flag to 1
Check sensor 2 charge time
Does sensor 2 charge time exceed tap threshold?
No
Yes
Set current loop flag to 1
previous sensor 1 flag is True AND current sensor 2 flag is True
Signal 1->2 sequence
previous sensor 2 flag is True AND current sensor 1 flag is True
Signal 1->2 sequence
Set all flags to 0
Set previous loop flag values to the corresponding current loop flag values

```
#include <ESP_CapSense.h>

const int sendPin = 4, leftReceivePin = 33, rightReceivePin = 15, LED = 13;
// resistor between pins 4 (A5) & 33, pin 33 is read pin
CapacitiveSensor leftFiber = CapacitiveSensor(sendPin, leftReceivePin);
// resistor between pins 4 (A5) & 33, pin 15 is read pin
CapacitiveSensor rightFiber = CapacitiveSensor(sendPin, rightReceivePin);

// timer variable [milliseconds]
unsigned long currentTime, startTime;
int waitTime = 5000; // 5 seconds

const int secondTapWait = 500;
const int tapDebounceTime = 100;

float leftFiberChargeTime, rightFiberChargeTime;

// threshold variables
int leftMinThreshold = 0, rightMinThreshold = 0, leftHoverThreshold = 0, leftTouchThreshold = 0, rightHoverThreshold = 0, rightTouchThreshold = 0;
float hoverFactor = 2, touchFactor = 5;

// program variables
int numberOfSamples = 30;
const int signalDelay = 100;
bool rightFiberDoubleTapFlag = false, leftFiberDoubleTapFlag = false, rightSwipeFlag = false, leftSwipeFlag = false;

/*********************************************/

void setup()
{
  leftFiber.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  rightFiber.set_CS_AutocaL_Millis(0xFFFFFFFF);  // turns off autocalibrate
  Serial.begin(115200);

  setupThresholds();
  setupLED();
}
```

```
/*************** HELPER FUNCTIONS ***************/

/**
 * Signals to user by flashing a LED fiber
 */
void signalToUser() {
  digitalWrite(LED, HIGH);
  delay(signalDelay);
  digitalWrite(LED, LOW);
}

/**
 * Initializes LED GPIO pin and sets the LED as off initially
 */
void setupLED() {
  pinMode(LED, OUTPUT);
  digitalWrite(LED, LOW);
}

/**
 * Throws away the first few seconds of data
 */
void setupThresholds() {
  currentTime = millis();
  while (millis() - currentTime < waitTime) {
    Serial.println("WAIT");
    Serial.print(leftFiber.capacitiveSensor(numberOfSamples));
    Serial.print(" ");
    Serial.println(rightFiber.capacitiveSensor(numberOfSamples));
  }

  for (int counter = 0; counter < 10; counter++) {
    leftMinThreshold += leftFiber.capacitiveSensor(numberOfSamples);
    rightMinThreshold += rightFiber.capacitiveSensor(numberOfSamples);
  }
  leftMinThreshold /= 10;
  rightMinThreshold /= 10;
  Serial.print("LEFT MIN THRESHOLD: ");
  Serial.println(leftMinThreshold);
  Serial.print("RIGHT MIN THRESHOLD: ");
  Serial.println(rightMinThreshold);

  rightHoverThreshold = hoverFactor * rightMinThreshold;
  rightTouchThreshold = touchFactor * rightMinThreshold;
  leftHoverThreshold = hoverFactor * leftMinThreshold;
  leftTouchThreshold = touchFactor * leftMinThreshold;
}
```

```
void loop()
{
  leftFiberChargeTime = leftFiber.capacitiveSensor(numberOfSamples);
  rightFiberChargeTime = rightFiber.capacitiveSensor(numberOfSamples);

  if (leftFiberChargeTime > leftTouchThreshold && rightFiberChargeTime < rightTouchThreshold)
{
    // Wait to allow user to withdraw their hand for another tap
    delay(tapDebounceTime);

    // start timer
    startTime = millis();

    while( (millis() - startTime) < secondTapWait) {
      leftFiberChargeTime = leftFiber.capacitiveSensor(numberOfSamples);
      rightFiberChargeTime = rightFiber.capacitiveSensor(numberOfSamples);

      if (leftFiberChargeTime > leftTouchThreshold) {
        leftFiberDoubleTapFlag = true;
        break;
      }
      if (rightFiberChargeTime > rightTouchThreshold) {
        rightSwipeFlag = true;
        break;
      }
    }
  }

  // checking flags
  if (rightSwipeFlag) {
    Serial.println("RIGHT SWIPE DETECTED");
    rightSwipeFlag = false;
  }
  if (leftFiberDoubleTapFlag) {
    Serial.println("LEFT DOUBLE TAP DETECTED");
    leftFiberDoubleTapFlag = false;
  }

}
```

Flip the logic from left fiber to right fiber!

```
void loop()
{
  leftFiberChargeTime = leftFiber.capacitiveSensor(numberOfSamples);
  rightFiberChargeTime = rightFiber.capacitiveSensor(numberOfSamples);

  if (leftFiberChargeTime > leftTouchThreshold && rightFiberChargeTime < rightTouchThreshold) {
    // Wait to allow user to withdraw their hand for another tap
    delay(tapDebounceTime);

    // start timer
    startTime = millis();

    while( (millis() - startTime) < secondTapWait) {
      leftFiberChargeTime = leftFiber.capacitiveSensor(numberOfSamples);
      rightFiberChargeTime = rightFiber.capacitiveSensor(numberOfSamples);

      if (leftFiberChargeTime > leftTouchThreshold) {
        leftFiberDoubleTapFlag = true;
        break;
      }
      if (rightFiberChargeTime > rightTouchThreshold) {
        rightSwipeFlag = true;
        break;
      }
    }
  }

  if (rightFiberChargeTime > rightTouchThreshold && leftFiberChargeTime < leftTouchThreshold) {
    // Wait to allow user to withdraw their hand for another tap
    delay(tapDebounceTime);

    // start timer
    startTime = millis();

    while( (millis() - startTime) < secondTapWait) {
      leftFiberChargeTime = leftFiber.capacitiveSensor(numberOfSamples);
      rightFiberChargeTime = rightFiber.capacitiveSensor(numberOfSamples);

      if (rightFiberChargeTime > rightTouchThreshold) {
        rightFiberDoubleTapFlag = true;
        break;
      }
      if (leftFiberChargeTime > leftTouchThreshold) {
        leftSwipeFlag = true;
        break;
      }
    }
  }
```

Then check flags

```
  // checking flags
  if (rightSwipeFlag) {
    Serial.println("RIGHT SWIPE DETECTED");
    rightSwipeFlag = false;
  }
  if (leftSwipeFlag) {
    Serial.println("LEFT SWIPE DETECTED");
    leftSwipeFlag = false;
  }
  if (rightFiberDoubleTapFlag) {
    Serial.println("RIGHT DOUBLE TAP DETECTED");
    rightFiberDoubleTapFlag = false;
  }
  if (leftFiberDoubleTapFlag) {
    Serial.println("LEFT DOUBLE TAP DETECTED");
    leftFiberDoubleTapFlag = false;
  }

}
```