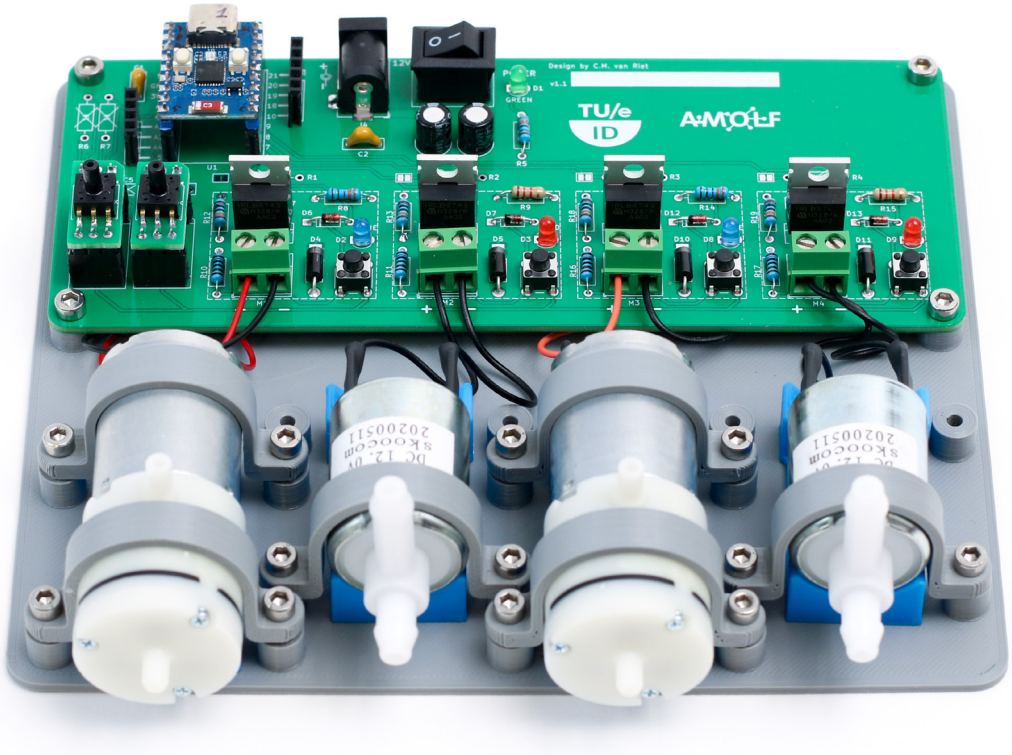
AIR CONTROL SYSTEM

MANUAL



C.M. van Riet - January '26 - v1.3

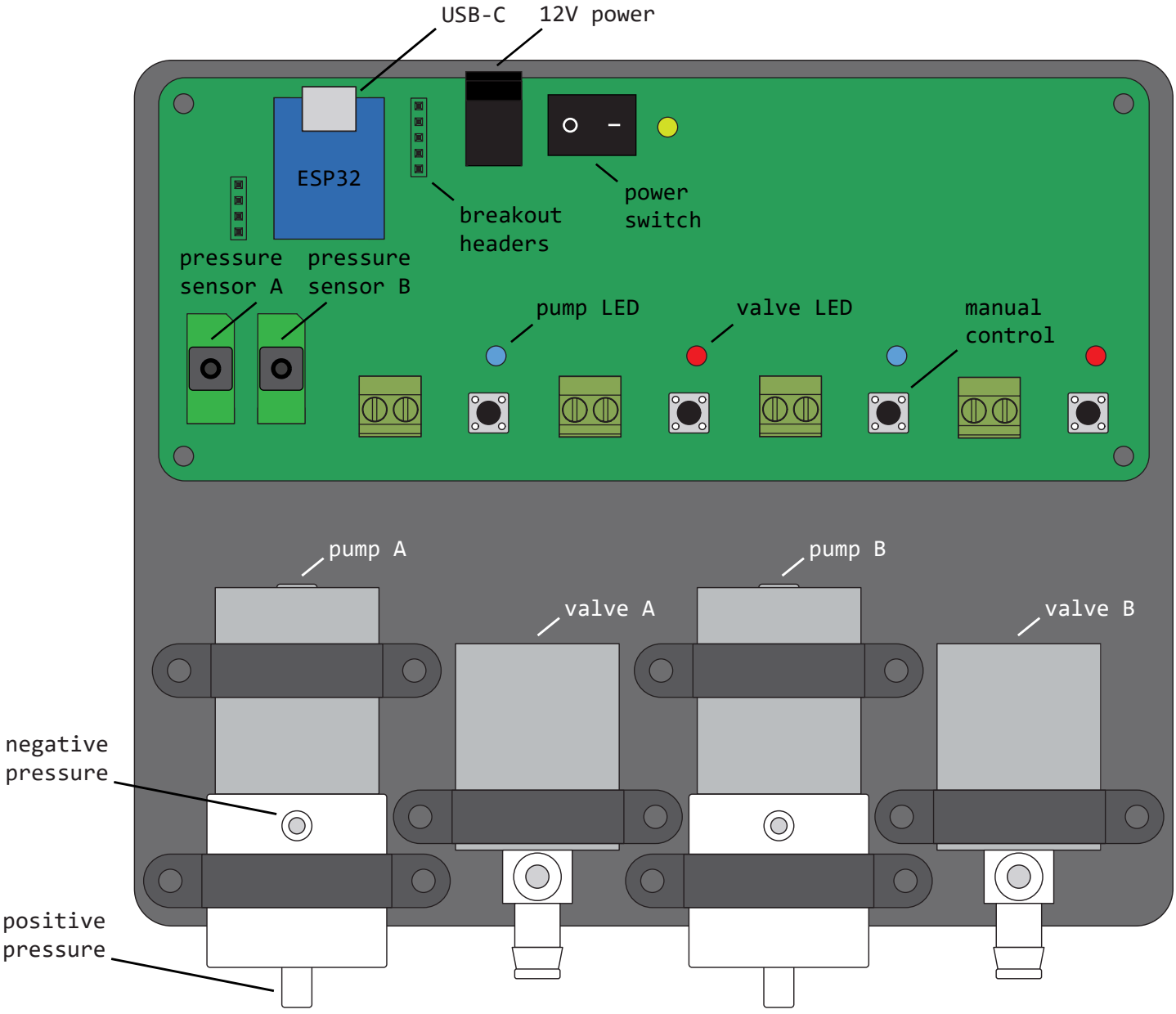
OVERVIEW

The Air Control System controls up to 4 pumps or valves. It is powered by a 12V power supply and an ESP32 C3 Zero, which can be programmed through the Arduino IDE.

In its standard configuration, each Air System contains two pumps, two solenoid valves, and two pressure sensors:

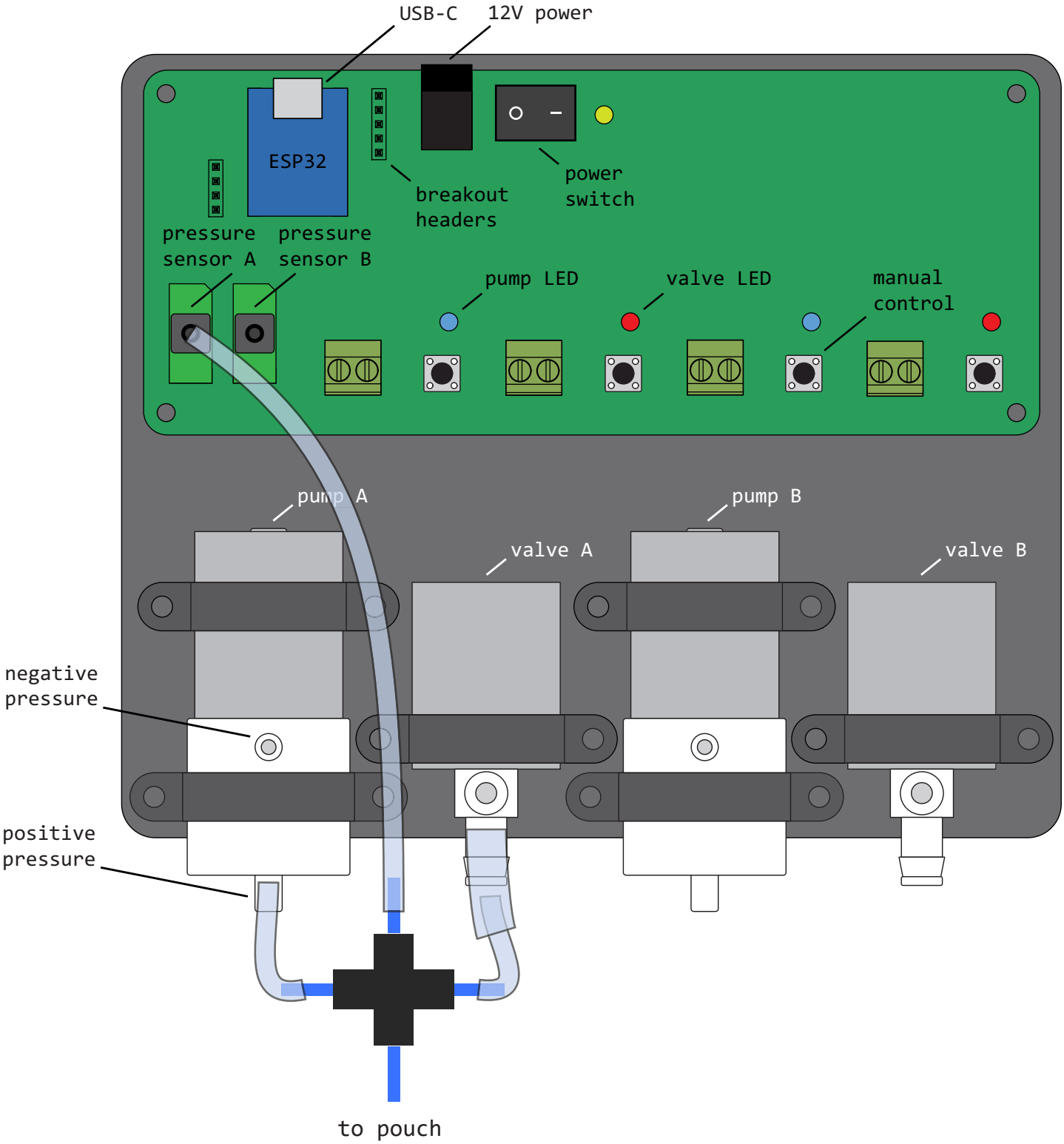
- The pump can provide positive and negative pressure (vacuum). You will mainly use positive pressure.
- The valve is normally closed. Activating the valve opens it.
- The pump and valve can be manually turned on by pressing the buttons.
- The LEDs indicate whether the pump or valve are turned on.

The pumps and valves can be replaced. In this way, the board can support any configuration of pumps and valves.



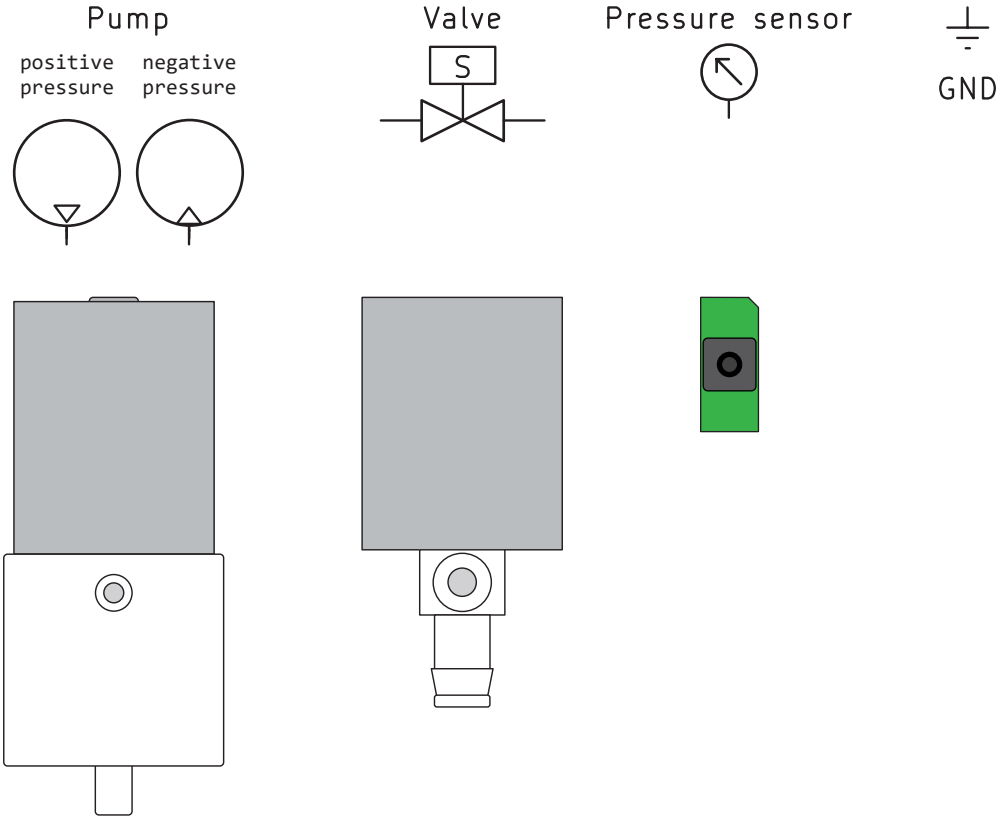
POUCH CONNECTION

To follow these steps, connect your pouch to the board as shown below.



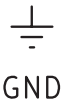
It can be helpful to draw out your pneumatic circuits when thinking about them.

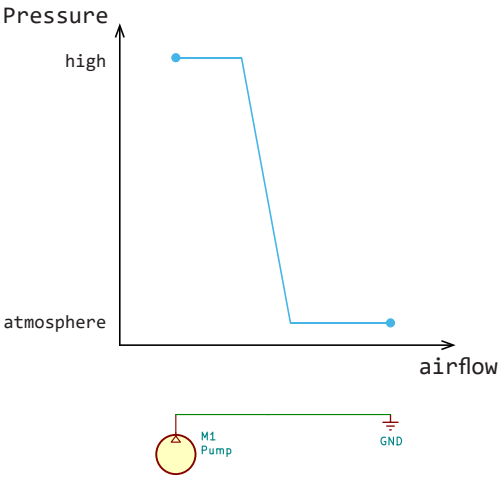
These are the symbols you can use:



Note: The atmospheric pressure around us is considered 'ground' (GND) pressure. The pump can create pressure compared to the 'ground' pressure of the atmosphere around us.

Any place in the circuit where air can leak out is considered 'ground'.



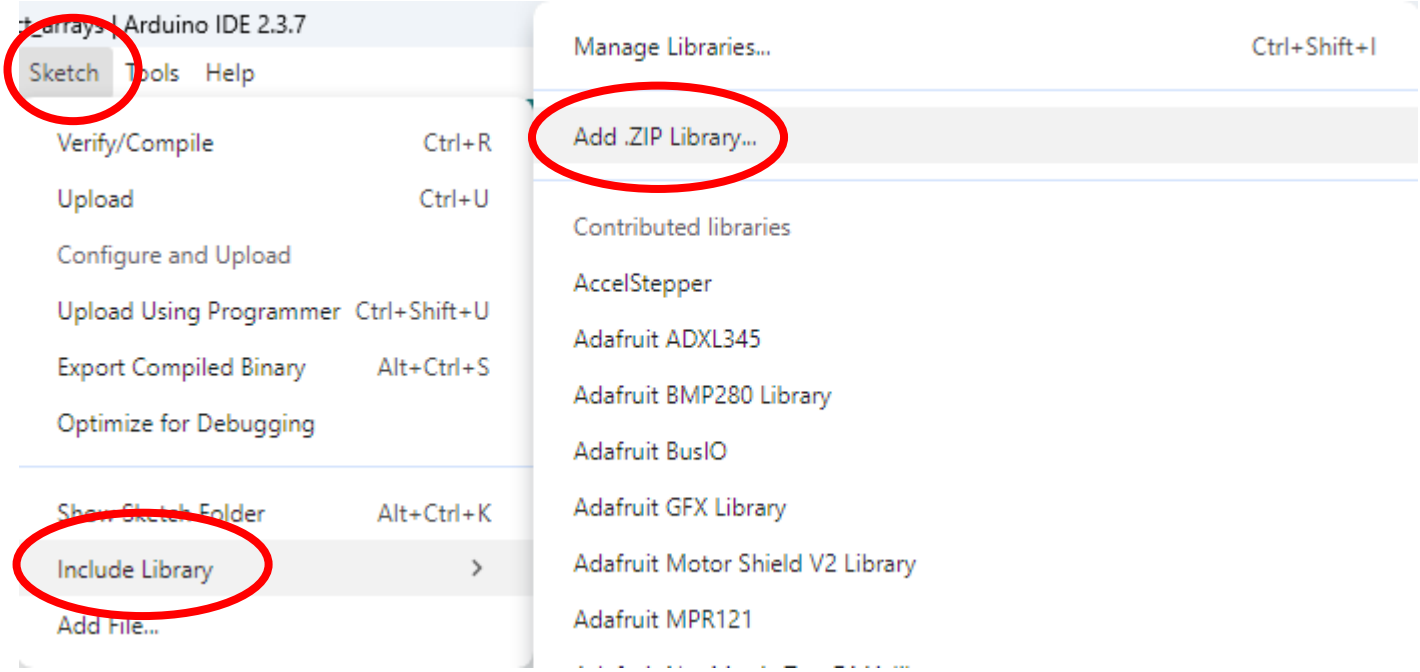


INSTALLING THE ARDUINO LIBRARY

Before you can make Arduino sketches using the AirSystem library, you need to install the library.

To do this, import the AirSystem-ESP folder (zipped) into your Arduino IDE.
You can download the zipped library here: https://github.com/kvriet/AirSystem-ESP32-for-Soft-Robotics/blob/main/AirSystem-ESP.zip

To install the library, start the Arduino IDE. Go to:
Sketch > Include Library > Add .ZIP Library.
Select the zipped library folder and click 'open'.
The Airsystem library should now be installed.



If everything went well, your Arduino library structure should look as follows:

```
Arduino/
    └── libraries/
        └── AirSystem-ESP/
            ├── AirSystem-ESP.h
            ├── AirSystem-ESP.cpp
            └── examples/
```

Whenever you want to use the library, don't forget to call it at the top of your sketch:

```
#include <AirSystem-ESP.h>
```

Also don't forget to initialise the pins in the *setup loop*:

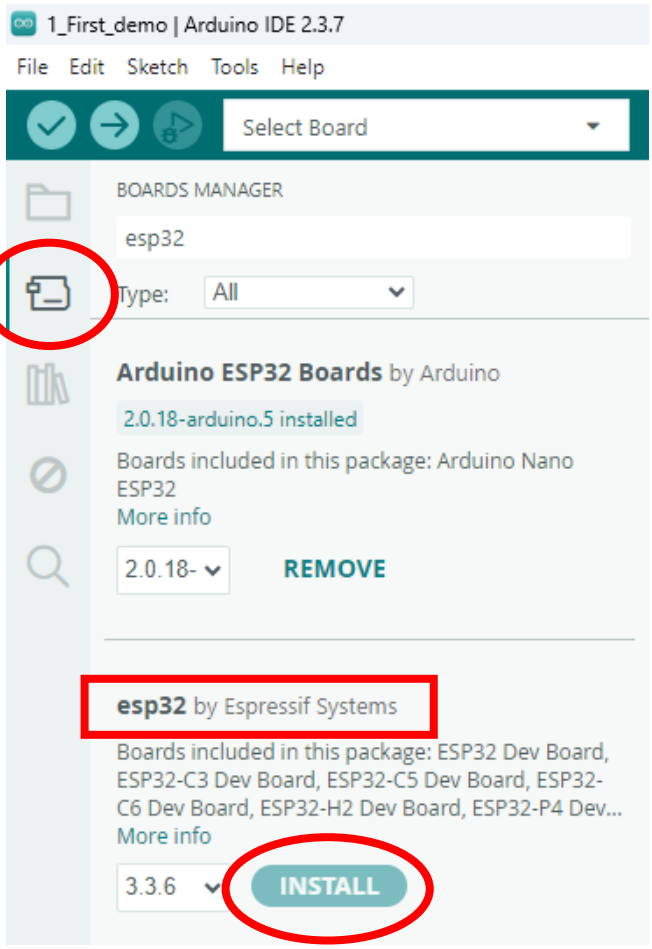
```
AirSystem_begin(); // Initialise the pins
```

INSTALLING THE ESP32 BOARD

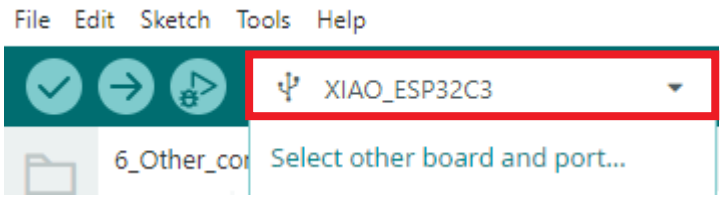
1. Connect the ESP32 to your computer.

2. In the Arduino IDE, go to File > Preferences...
   In the 'Additional boards manager URLs' box, paste this URL:
   https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

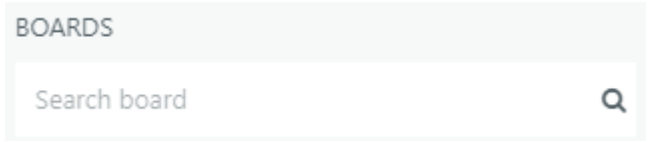
3. Click the board library. Search for 'ESP32'.
   Install the ESP32 Espressif board library.
   This will take a moment.



4. Click the USB connection drop-down menu (circled in red)
   Click 'Select other board and port...'



Search for 'XIAO ESP32 C3'



Select 'XIAO_ESP32C3' and click the correct port on the right of the menu.

You are now done and able to upload sketches to the ESP32!

PROGRAMMING

The **AirSystem** library is an object-oriented library that provides three types of objects:
- **Pump** → controls an air pump using PWM (0–100% power).
- **Valve** → controls a solenoid valve (open/close).
- **Sensor** → reads air pressure from a pressure sensor.

Accessing Objects

You can access objects in two ways:
By **name** (if you created them individually):

```
- PumpA.set(50);            // Set PumpA to 50% power
- ValveB.open();            // Open ValveB
- float p = SensorA.read();  // Read SensorC pressure
```

By **array** index:

```
- motors[0].set(50);        // PumpA
- motors[1].on();           // ValveB
- motors[2].off();          // PumpB
```

Functions

Pump/Motor

- set(percentage) → Set pump power (0–100%).
- off() → Turn pump off (same as set(0)).
- on() → Turn pump fully on (100%).

Valve

- open() → Open the valve.
- close() → Close the valve.

Sensor

- read() → Returns the current pressure in kPa (kiloPascal).

STEP 1: MANUAL CONTROL

(All these steps are also included as examples in the Airsystem library.)

The pump and valves can be operated through the push buttons on the follower unit.

Upload the following code. It will not activate the pumps or valves. Instead you can press the pump and valve buttons to inflate and deflate your pneumatic button.

Turn on the Airsystem with the power switch. The green LED should light up.
If you quickly want to deactivate your pumps and valves, you can always switch off the power.

```
/*
   AirSystem Demo Sketch
   ---------------------

   Author: C.M. van Riet
   Library: AirSystem-ESP
   Date:   January 2026

  Objects can be called in 3 ways:
     - pumpA, valveB, sensorA (named)
     - M1, M2, M3, M4 (named)
     - pumps[0], valves[1], sensors[0] (array-based)

   Functions available in this library:
     - Initialize the system with AirSystem_begin()
     - Control pumps with .set(percentage), on(), and .off()
     - Control valves with .open() and .close()
     - Read pressure from sensors with .read()

   Hardware:
   - Designed for a maximum of 4 pumps or 4 valves, and 2 sensors
   - Pin mappings are fixed in the library (see AirSystem-ESP.h)

   Notes:
   - Make sure to include the AirSystem-ESP.h library at the top of your sketch
     (#include <AirSystem-ESP.h>)
   - This example can be found under File > Examples > AirSystem-ESP > 1_First_demo.
*/

#include <AirSystem-ESP.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();       // Pump off
  valveA.close();    // Close valve
  pumpB.off();       // Pump off
  valveB.close();    // Close valve
}

void loop() {
  // This loop is empty
  // You can turn on the pump and valve by pressing
  // the buttons on the Air System board
}
```

STEP 2: PRESSURE READINGS

To get the pressure sensor value, use the read() function. The pressure is in kPa.

Upload the following code. Use the push buttons to inflate and deflate your pouch and use the Serial Monitor and Serial Plotter (top right of IDE) to watch the pressure value fluctuate.


```
#include <AirSystem-ESP.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();       // Pump off
  valveA.close();    // Close valve
  pumpB.off();       // Pump off
  valveB.close();    // Close valve
}

void loop() {
  float pressure = sensorA.read();
  Serial.print(0);     // Lower limit in Serial Plotter
  Serial.print("\t");  // Print a tab
  Serial.print(200);   // Upper limit in Serial Plotter
  Serial.print("\t");  // Print a tab
  Serial.println(pressure);

  delay(10);
}
```

STEP 3: PUMP CONTROL

The code below turns the pump on and off.

- Try to change the delay values. How does the pump turning on and off change?
- Try to change the value in the set() function. How does it change the pump behaviour?
- When not activated, does the pump allow air to leak out?

```
#include <AirSystem-ESP.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();       // Pump off
  valveA.close();    // Close valve
  pumpB.off();       // Pump off
  valveB.close();    // Close valve
}

void loop() {
  pumpA.set(50);
  Serial.println("Pump set to 50% intensity");

  delay(2000);

  pumpA.off();
  Serial.println("Pump turned off");

  delay(2000);
}
```

STEP 4: VALVE CONTROL

The code below is similar to the code in step 3, but it adds valve control. Now you can inflate and deflate your pouch. Above 100kPa, the solenoid valve will release air, but not break.

- Play with the pump power percentage and the timing (delay) to get an aesthetically pleasing inflation and deflation of your pouch.

```
#include <AirSystem-ESP.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();       // Pump off
  valveA.close();    // Close valve
  pumpB.off();       // Pump off
  valveB.close();    // Close valve
}

void loop() {
  valveA.close();
  pumpA.set(25);
  Serial.println("Valve closed and pump set to 25%");

  delay(1000);

  valveA.open();
  pumpA.off();
  Serial.println("Valve opened and pump turned off");

  delay(2000);
}
```

STEP 5: PUMP WITHOUT DELAY

In step 4 we turned the pump on and off. We used the delay() function for this, but this freezes up the entire program; you can't do anything else during the delay. To fix this, the code below gives pump control without delay, similar to the 'Blink without delay' example in the Arduino library. For this to work, we use a general timer (currentMillis) to keep track of time.

```
// Pump without delay based on the code by David A. Mellis

#include <AirSystem-ESP.h>

// Variables will change:
int pumpState = LOW;  // pumpState used to set the pump on/off

// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0;  // will store last time pump state was updated

// Constants won't change:
const long interval = 1000;  // interval at which to blink (milliseconds)
// Blink here means turn the pump on and off

#include <AirSystem-ESP.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();        // Pump off
  valveA.close();     // Close valve
  pumpB.off();        // Pump off
  valveB.close();     // Close valve
}

void loop() {
  // Here is where you'd put code that needs to be running all the time.

  // Get the current elapsed milliseconds from the start of the sketch
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
    // This if-statement checks if the current time is bigger than the previous time - interval
    // So check if the interval time (1 second) has elapsed
    previousMillis = currentMillis;
    // If the pump is off, turn it on and vice-versa:
    if (pumpState == LOW) {
      pumpState = HIGH;
      pumpA.set(50);   // Set pump to 50%
      valveA.close();  // Close valve
    } else {
      pumpState = LOW;
      pumpA.off();     // Turn pump off
      valveA.open();  // Open valve
    }
  }
  Serial.print(currentMillis);
  Serial.print("\t");
  Serial.print(previousMillis);
  Serial.print("\t");
  Serial.printn(interval);
}
```

STEP 6: OTHER CONFIGURATIONS

In the previous examples, we assumed that the board layout contained 2 pumps and 2 valves in a pump, valve, pump, valve configuration. However, if you want to use different configurations, we need more generic names to call these components. For this we use the 'motor' designation: M1, M2, M3, M4. This does not affect how the sensors are called.
Because Arduino does not know what type of component it is addressing, control functions are limited to .set(percentage), on(), and .off()

```
#include <AirSystem-ESP.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin();  // Initialise the pins
  M1.off();       // Component 1 off
  M2.off();       // Component 2 off
  M3.off();       // Component 3 off
  M4.off();       // Component 4 off
}

void loop() {
  // Set component 3 to 25% (this is pump B) and component 4 to 10% (this is valve B)
  M3.set(25);
  M4.set(10);   // Notice that at 10%, valve B does not open!
  delay(1000);

  // Set component 3 to 0% and component 4 to 50%
  M3.set(0);
  M4.set(50);   // Notice that at 50%, valve B does open!
  delay(1000);

  float pressure = sensorA.read();  // Calling sensors remains unaffected
  Serial.println(pressure);
}
```

STEP 7: OBJECT ARRAYS

In the previous examples we worked with named objects for the pump and valve. However, you can also directly address these objects within they object array. This can be useful if you want to loop through the objects. The code below shows you how to access pumps and valves through motors[i].

```
#include <AirSystem-ESP.h>

int motorPercentage = 0;

void setup() {
  Serial.begin(9600);
  AirSystem_begin();  // Initialise the pins
  for (int i = 0; i < 4; i++) {
    // Turn off all components
    motors[i].off();
  }
}

void loop() {
  motors[0].set(motorPercentage); // pump A
  motors[1].set(motorPercentage); // valve A

  motorPercentage++;  // Add 1 to the motorPercentage variable

  if (currentMillis > 100) {
    motorPercentage = 0; // If motorPercentage becomes larger than 100, make it 0 again
  }

  delay(100);
}
```

PIN OVERVIEW (STANDARD CONFIGURATION)

```
PumpA = 6
PumpB = 8

ValveA = 7
ValveB = 9

SensorA = 0
SensorB = 1

LED_BUILTIN = 10 (WS2812, RGB)
```