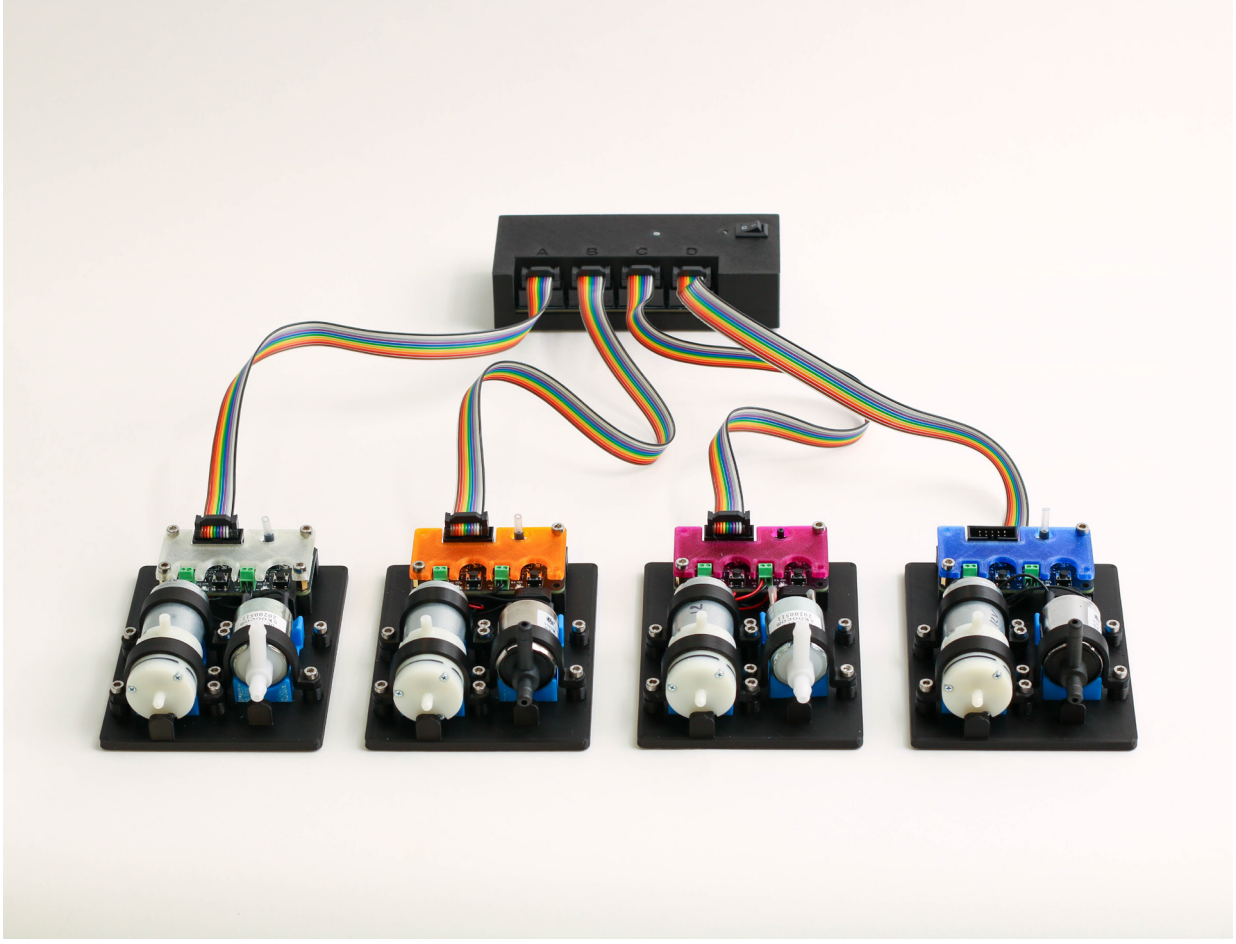
AIR CONTROL SYSTEM

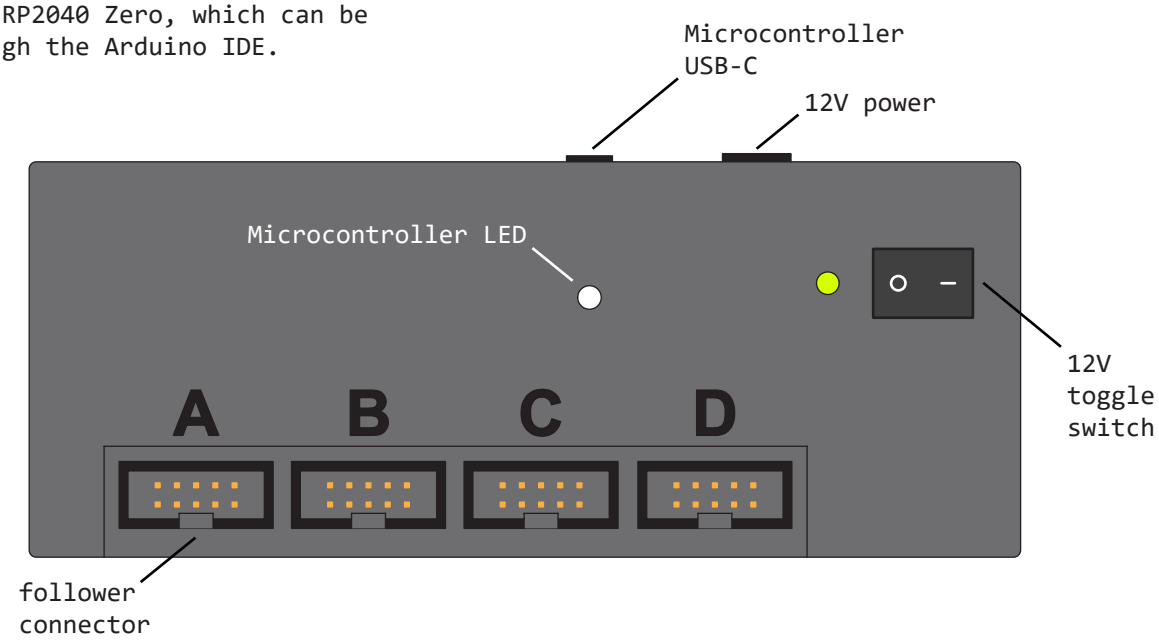
MANUAL



C.M. van Riet - September '25 - v1.3

OVERVIEW

Controller
The controller can control up to 4 pneumatic units. It is powered by a 12V power supply and a Waveshare RP2040 Zero, which can be programmed through the Arduino IDE.



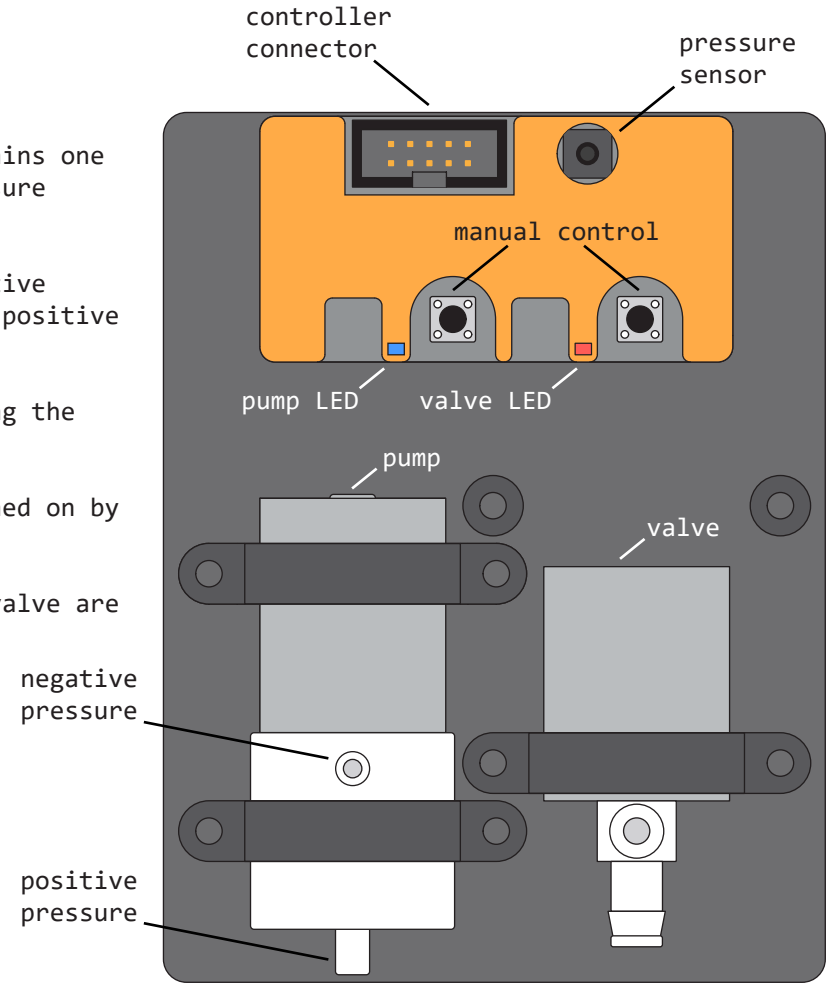
Pneumatic unit
Each pneumatic unit (or follower) contains one pump, one solenoid valve, and one pressure sensor.

The pump can provide positive and negative pressure (vacuum). You will mainly use positive pressure for the pneumatic buttons.

The valve is normally closed. Activating the valve opens it.

The pump and valve can be manually turned on by pressing the buttons.

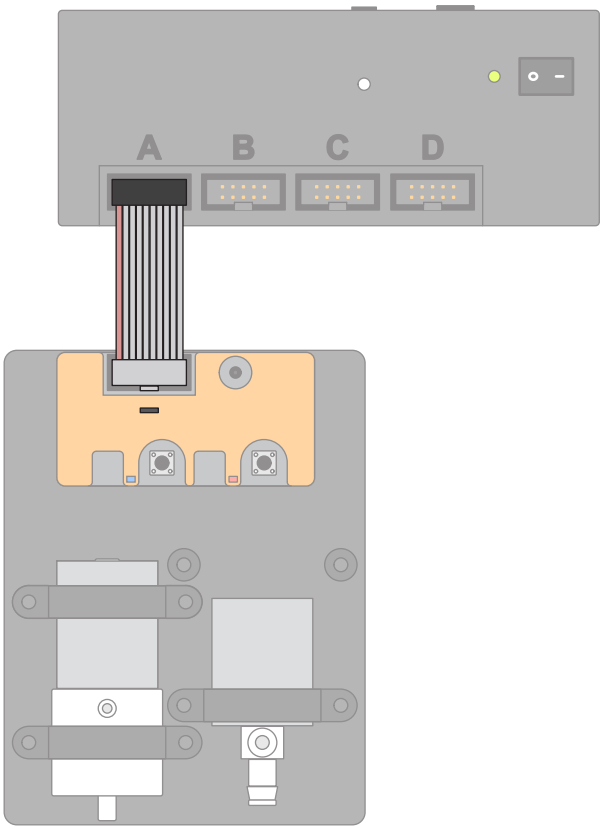
The LEDs indicate whether the pump or valve are turned on.



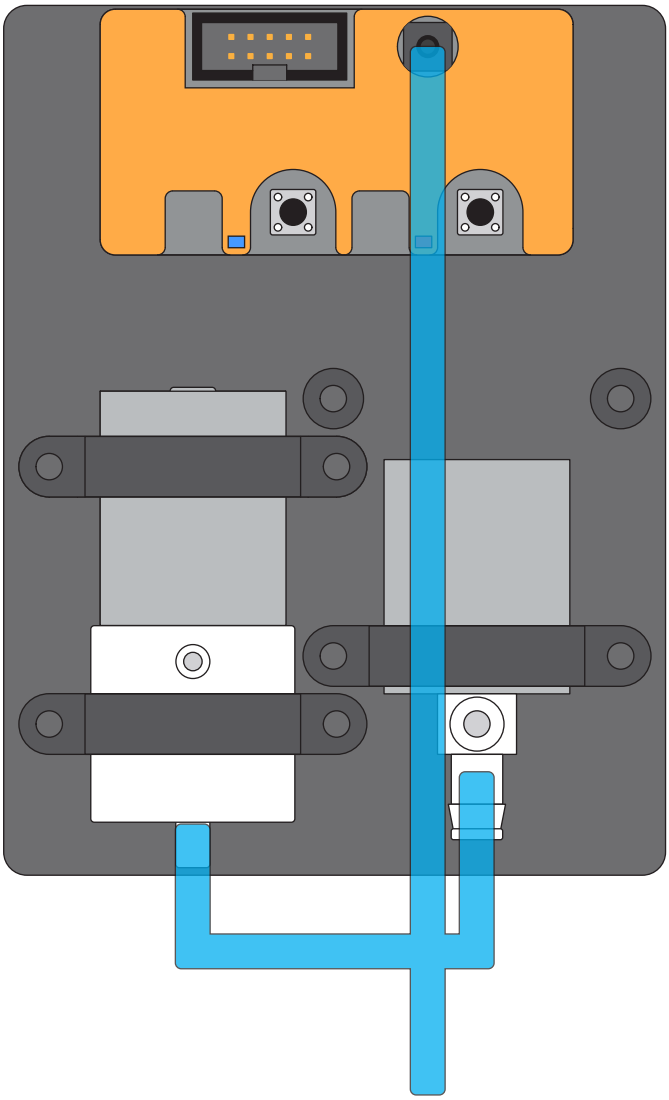
Units are connected by ribbon cables. Each controller can connect up to 4 units.

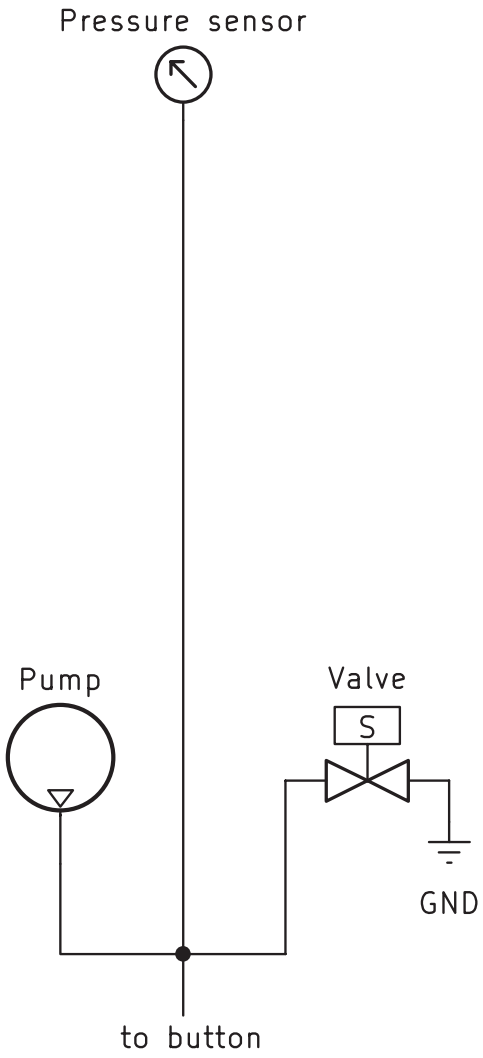
The black end of the ribbon cable goes into the controller. The grey end of the ribbon cable goes into the pneumatic unit.

NOTE: disconnect the USB cable before inserting or removing ribbon cables!



SCHEMATICS





INSTALLING THE ARDUINO LIBRARY

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

To do this, put the AirSystem folder (unzipped) into your Arduino *libraries* folder:

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

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

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

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

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

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 = SensorC.read();  // Read SensorC pressure
```

By **array** index:

```
- pumps[0].set(50);           // PumpA
- valves[1].close();          // ValveB
- float p = sensors[2].read(); // SensorC
```

Functions

Pump

```
- 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

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.

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

   Author: C.M. van Riet
   Library: AirSystem
   Date:   September 2025

  Objects can be called in 2 ways:
     - pumpA, valveB, sensorC (named)
     - pumps[0], valves[1], sensors[2] (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, 4 valves, and 4 sensors
   - Pin mappings are fixed in the library (see AirSystem.h)

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

#include <AirSystem.h>

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

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

STEP 2: PRESSURE READINGS

To get the pressure sensor value, use the read() function.

Upload the following code. Use the push buttons to inflate and deflate your pneumatic button and use the Serial Monitor (Ctrl+Shift+M) and Serial Plotter to watch the pressure value fluctuate.

```
#include <AirSystem.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();       // Pump off
  valveA.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. Don't connect the pump to your pneumatic button yet!

- 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?
- If you connect the pump to your pneumatic button, will it only inflate? Or also deflate?

```
#include <AirSystem.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();       // Pump off
  valveA.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 your pneumatic button will inflate and deflate.

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

```
#include <AirSystem.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();       // Pump off
  valveA.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 you pump control without delay. This is 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.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 LED 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.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin(); // Initialise the pins
  pumpA.off();        // Pump off
  valveA.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: HAPTIC EFFECT

Now that we are familiar with the basics of the Air System, we can start playing with haptic effects. In the code below, the pump produces a certain target pressure (setpoint). When you press your pneumatic button, and you reach a certain threshold pressure, the valve suddenly opens, thereby producing a snapping effect in the button.

```
#include <AirSystem.h>

int setpoint = 30;            // The target pressure (kPa) for the button
int offset = 2;               // Optional: how much higher the target pressure can be before
                              // pump turns off (to prevent pump jitter)
int threshold = 45;           // The pressure at which the buttons 'snaps' down
int counter_number = 3;       // For how many loops the button empties
int counter = 0;              // We start our haptic effect counter at 0
bool doHapticEffect = false;  // Currently don't do the haptic effect yet

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

void loop() {
  float pressure = sensorA.read();

  // Pump control
  if (doHapticEffect == false) {
    // If we're not doing the haptic effect:
    if (pressure < setpoint) {
      // If the pressure is not at the target level, keep pumping
      pumpA.set(100);  // Set pump to 100%
      valveA.close();  // Keep valve closed
    } else if (pressure >= setpoint + offset) {
      // If target pressure has been reached (+ offset value), turn pump off
      pumpA.off();
    }
  }

  if (pressure > threshold) {
    // If the pressure meets haptic effect threshold, set doHapticEffect to true
    doHapticEffect = true;
  }

  if (doHapticEffect == true) {
    // If we have to do the haptic effect, turn off pump and open valve
    pumpA.off();
    valveA.open();
    counter++;  // Increment counter
    if (counter > counter_number) {
      // After 3 cycles, the haptic effect resets and pressure will need to meet threshold again
      doHapticEffect = false;
      counter = 0;
    }
  }
  delay(10);
}
```

STEP 7: OBJECT ARRAYS

In the previous examples we worked with named objects for the pump, valve, and sensor. 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: pumps[i]
- Valves: valves[i]
- Sensors: sensors[i]

```
#include <AirSystem.h>

void setup() {
  Serial.begin(9600);
  AirSystem_begin();  // Initialise the pins
  pumps[0].off();     // Pump A off
  valves[0].close();  // Close valve A
}

void loop() {
  // Close valve A and set pump A to 25%
  valves[0].close();
  pumps[0].set(25);
  delay(1000);

  // Get pressure sensor A value and print
  float pressure = sensors[0].read();
  Serial.println(pressure);

  // Open valve A and turn pump A off
  valves[0].open();
  pumps[0].off();
  delay(2000);
}
```

PIN OVERVIEW

```
PumpA = 15
PumpB = 13
PumpC = 11
PumpD = 9

ValveA = 14
ValveB = 12
ValveC = 10
ValveD = 8

SensorA = 29
SensorB = 28
SensorC = 27
SensorD = 26

LED_BUILTIN = 16 (WS2812, RGB)
```