```
# Raspberry Pico version of Make: Electronics alarm

from machine import Pin
from time import sleep

# LED outputs
l1 = Pin(17, Pin.OUT) # Switches are closed GP17
l2 = Pin(16, Pin.OUT) # Time to leave       GP16
l3 = Pin(14, Pin.OUT) # Alarm is triggered  GP14
l4 = Pin(15, Pin.OUT) # Alarm!              GP15

# Inputs with pulldown
go = Pin(8, Pin.IN, Pin.PULL_DOWN)       # Go button
sensors = Pin(20, Pin.IN, Pin.PULL_DOWN) # Sensor switches

# show sensor state on LED 1
l1.value(sensors.value())

# function to be called when the sensor pin changes
# show sensor state on LED 1
def sensors_change(p):
    l1.value(sensors.value())

# connect function to the sensor pin change interrupt
sensors.irq(trigger=Pin.IRQ_RISING | Pin.IRQ_FALLING,
            handler=sensors_change)

# The alarm logic starts here

# Wait for GO button press
while go.value() == 0:
    pass

l2.value(1) # time to leave
sleep(10)   # exit delay
l2.value(0)

# wait for sensors to open
while sensors.value() == 1:
    pass

l3.value(1) # alarm is triggered
sleep(10)   # last chance delay
l3.value(0)

l4.value(1) # Alarm!

# the alarm stays on until reset/power-off
while True:
    sleep(1)
```

Inputs:

Alarm sensors. GP20 - Pin 26. Normally closed switches in alarm circuit pull to V+

Go button GP8 - Pin 11. Momentary pushbutton, connects input to V+

Reset button - Pin 30. Pull low to reset or turn off power to disable the alarm. It would be annoying to disconnect the power if the pico is powered with a USB cable.

Outputs:

LED 1 Switches are closed "Ready"	GP17 - Pin 22
LED 2 Time to leave	GP16 - Pin 21
LED 3 Alarm is triggered	GP14 - Pin 19
LED 4 Alarm!	GP15 - Pin 20

The LED4 pin can also be used to turn on an alarm.
Alternatively one could generate an audio frequency on another pin,
and feed to a speaker. Or a pulsing voltage for an intermittent alarm.

Program logic:

```
1 set up input and output pins
2 wait for GO button press
3 turn on LED 2 "Time to leave"
exit delay (sensor switches ignored)
turn off LED 2
4 wait for sensor switches to open
5 turn on LED 3 "Alarm is triggered"
last chance delay
turn off LED 3 "Alarm is triggered"
6 turn on LED 4
sound alarm
```

Wiring:

Half breadboard with dual buses. The red bus is connected to the 3.3V output
of the Pico. The circuit can be powered over USB.

The reset button is white, the go button is black.
The slide switch on the right is the sensors switch,
slider to the right means sensors closed.

The code:

Install Micropython on the Pico,
https://www.raspberrypi.com/documentation/microcontrollers/micropython.html#drag-and-drop-micropython
Copy main.py to the Pico. With that name the code runs when the board powers up.