Supplementary Information

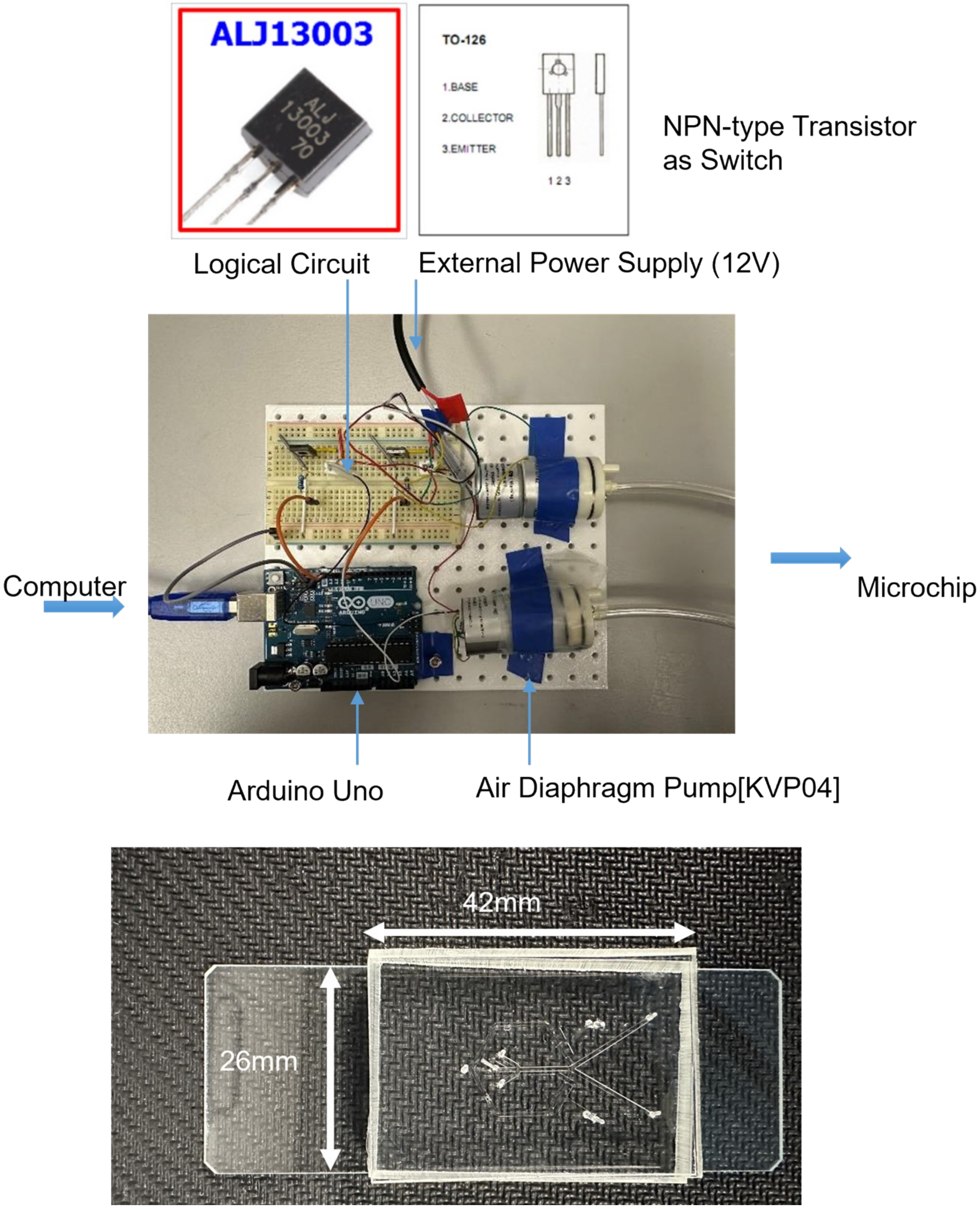

Figure.S1 The sorting logarithm hardware circuit and image of whole chip

Figure S1 illustrates the external control circuit, which comprises an Arduino development board, a breadboard-based control switch, an air diaphragm pump (KVP04), and an external 12V power supply. The circuit is connected to a superior computer via a data cable and linked to the microfluidic chip through a tube and a pneumatic-hydraulic conversion device. The primary component of the control circuit is the NPN-type transistor, which functions as a switch.

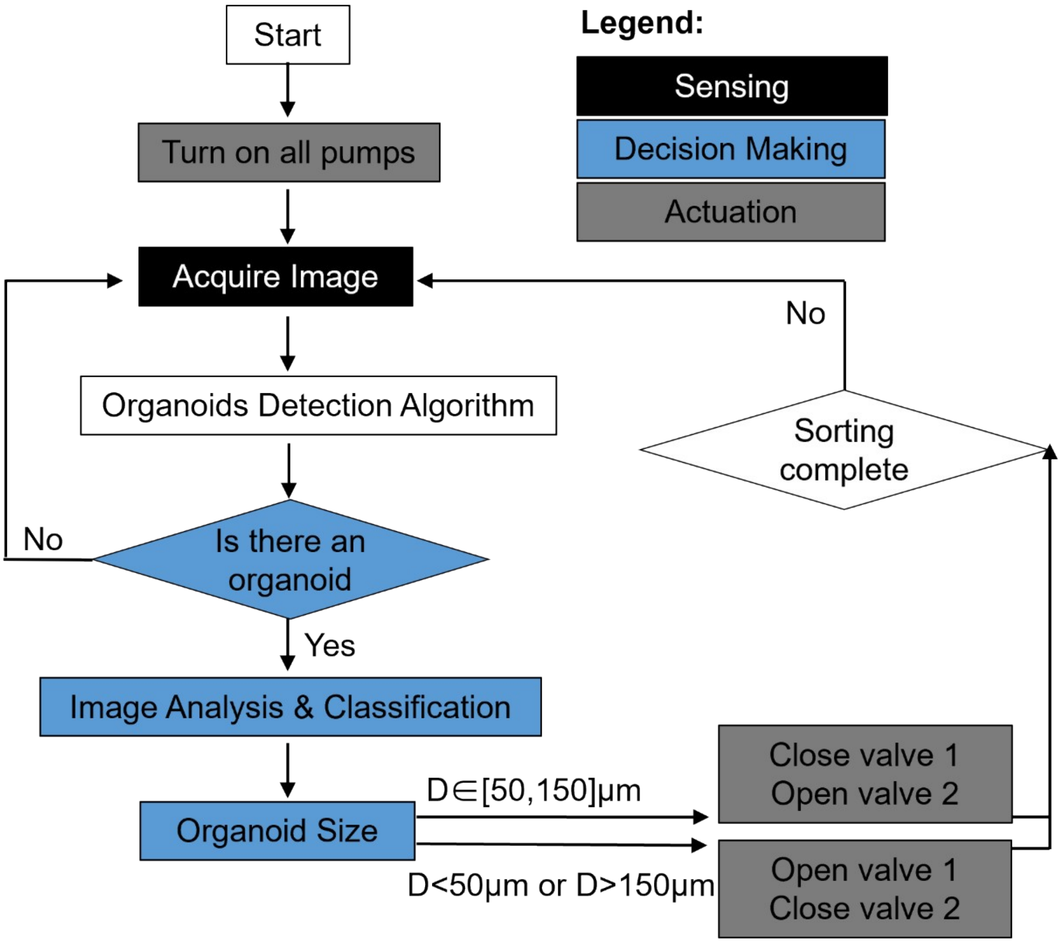

Figure.S2 The sorting logarithm in Python & Arduino

The sorting algorithm, executed in Python and Arduino, is depicted in Figure S2 and comprises several steps: 1) Capturing an image of the microfluidic channel; 2) Detecting an organoid within the captured image; 3) Conducting image processing and classification if an organoid is identified; 4) Actuating sorting valves subsequent to classification.

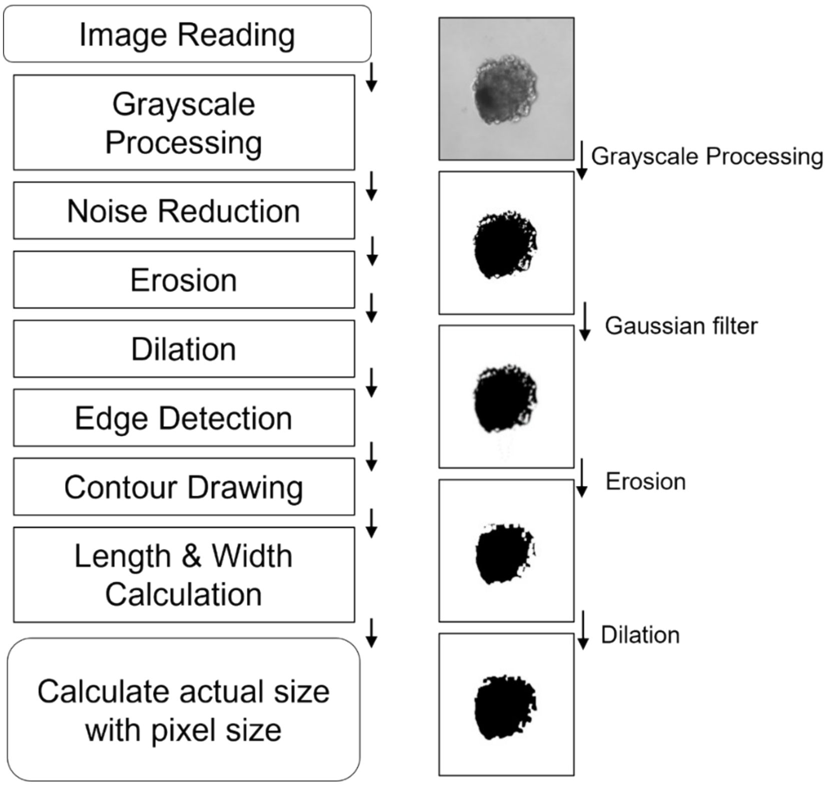

Figure.S3 Image Recognition Process of sorting process and Python codes below.

```
# Grayscale Processing
gray = cv2. cvtColor(img, cv2.COLOR_BGR2GRAY)
# Gaussian filter
kernel = np.ones((5,5), np.float32) /25
dst = cv2. filter2D(img, -1, kernel)
# Erosion: removes thin lines
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (9,9))
dst1 = cv2. erode(gray, kernel, iterations=15)
# Dilation: fills in small holes
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (9,9))
dst2 = cv2. dilate(dst1, kernel, iterations=15)
# Edge Detection: morphology gradient
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
dst = cv2. morphologyEx(gray, cv2.MORPH_GRADIENT, kernel, iterations=1)
# Find contours
result, contours, h = cv2.findContours(close, cv2. RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
# Define thresholds for detecting length and width
# Relatively according to the mangification rate
min_w = 10
min_h = 10
max_w = 20
max_h = 20
```