THE INTERNATIONAL JOURNAL OF DEVELOPMENTAL BIOLOGY
www.intjdevbiol.com

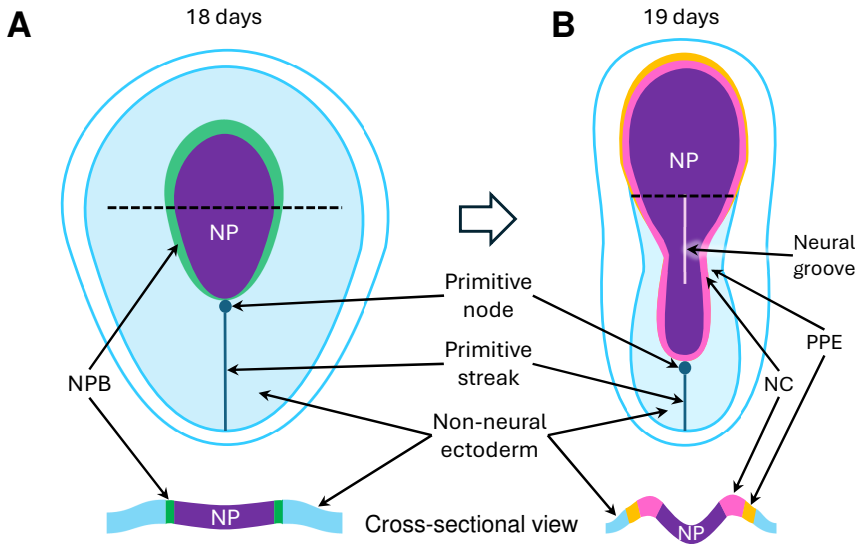
Supplementary Material

corresponding to:

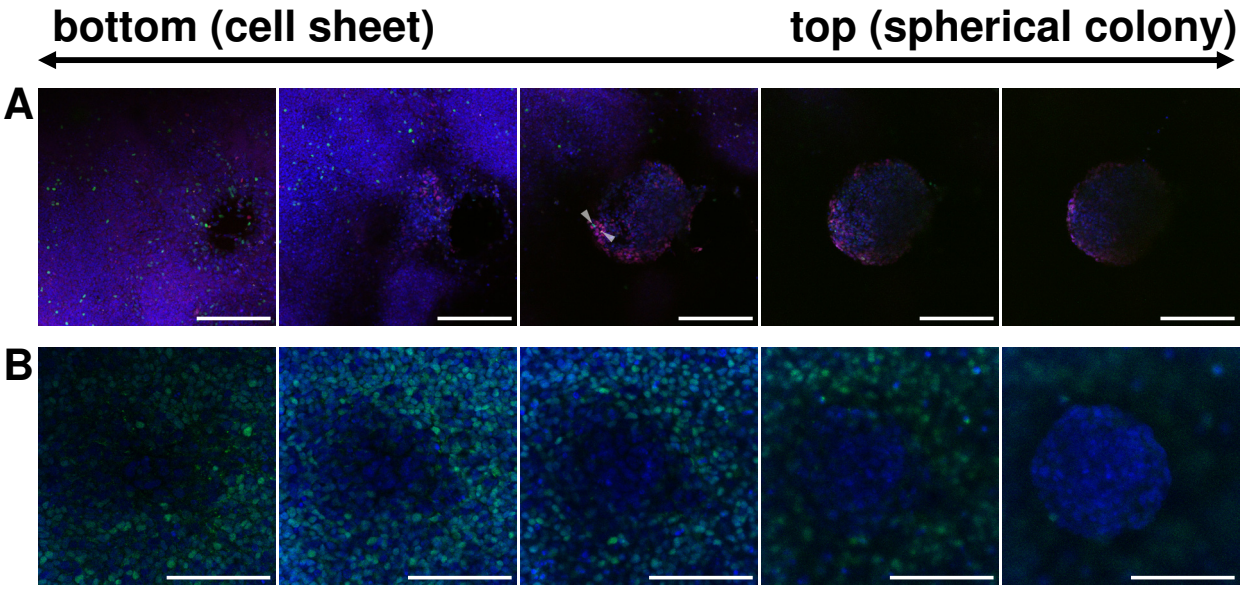
Tension-induced enhancement of SIX1 expression during preplacodal ectoderm differentiation from human induced pluripotent stem cells

SEUNGTAE KIM, AYUMI HORIKAWA, TAKAYOSHI YAMAMOTO, TATSUO MICHIUE

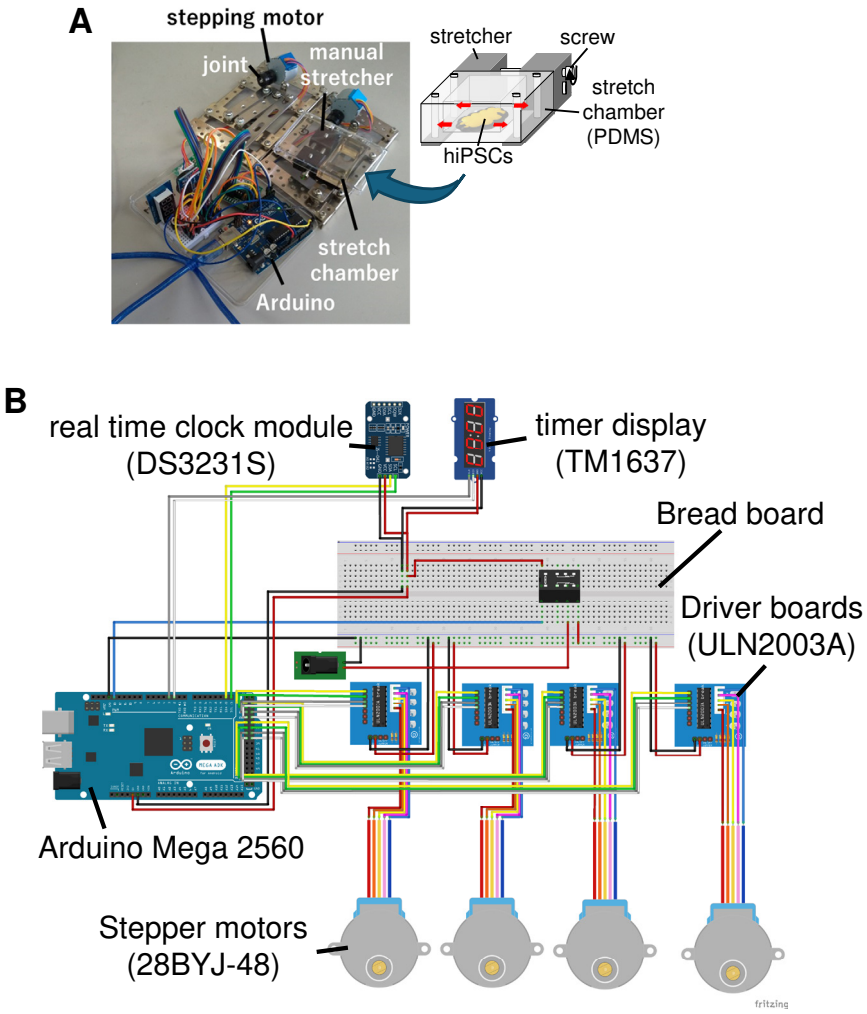
Full Text for this paper is available at: https://doi.org/10.1387/ijdb.240212tm



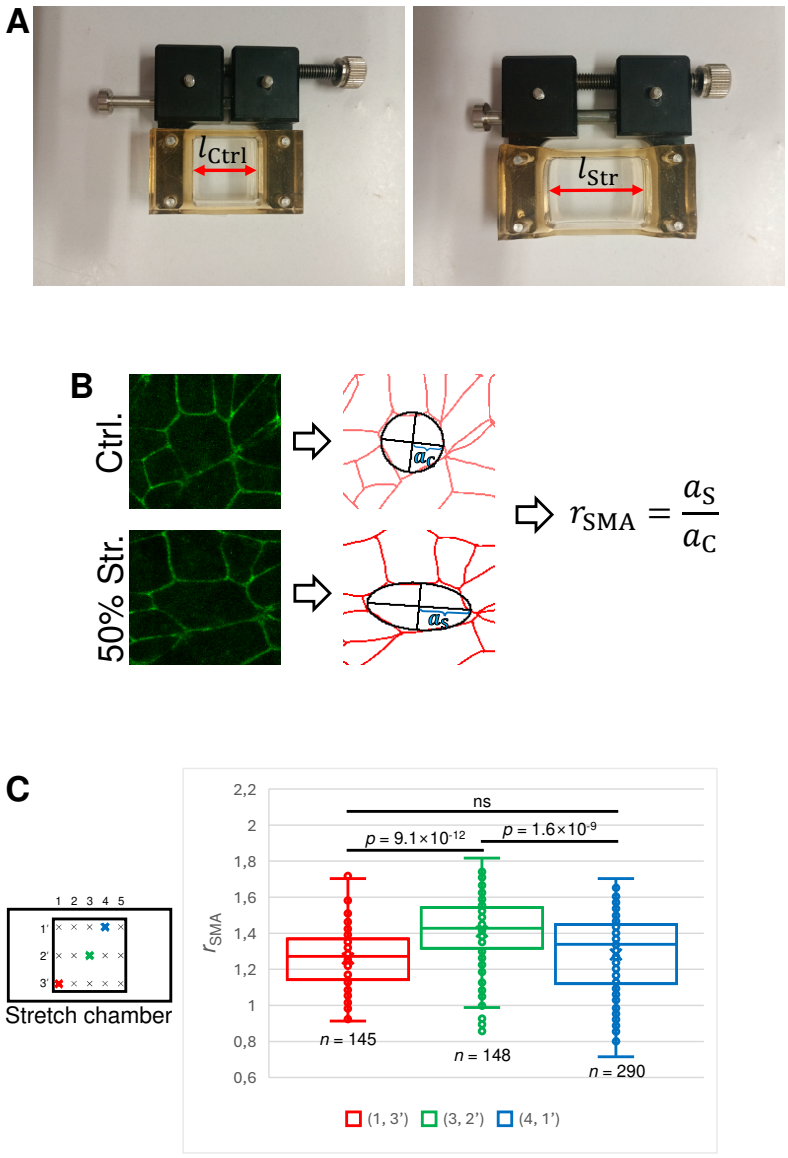
Supplementary Figure 1. Schematic diagram of early neural development in amniote. Dorsal view of amniote embryos from late gastrula **(A**, 18 days**)** to early neurula **(B**, 19 days**)** (Esmaeli *et al.*, 2024; Sadler, 2005; Sen and Jangra, 2023). The most outside line represents a cut edge of amnion. The dashed lines stand for the region of the cross-sectional views shown below. NP: neural plate; NPB: neural plate border; NC: neural crest; PPE: preplacodal ectoderm.



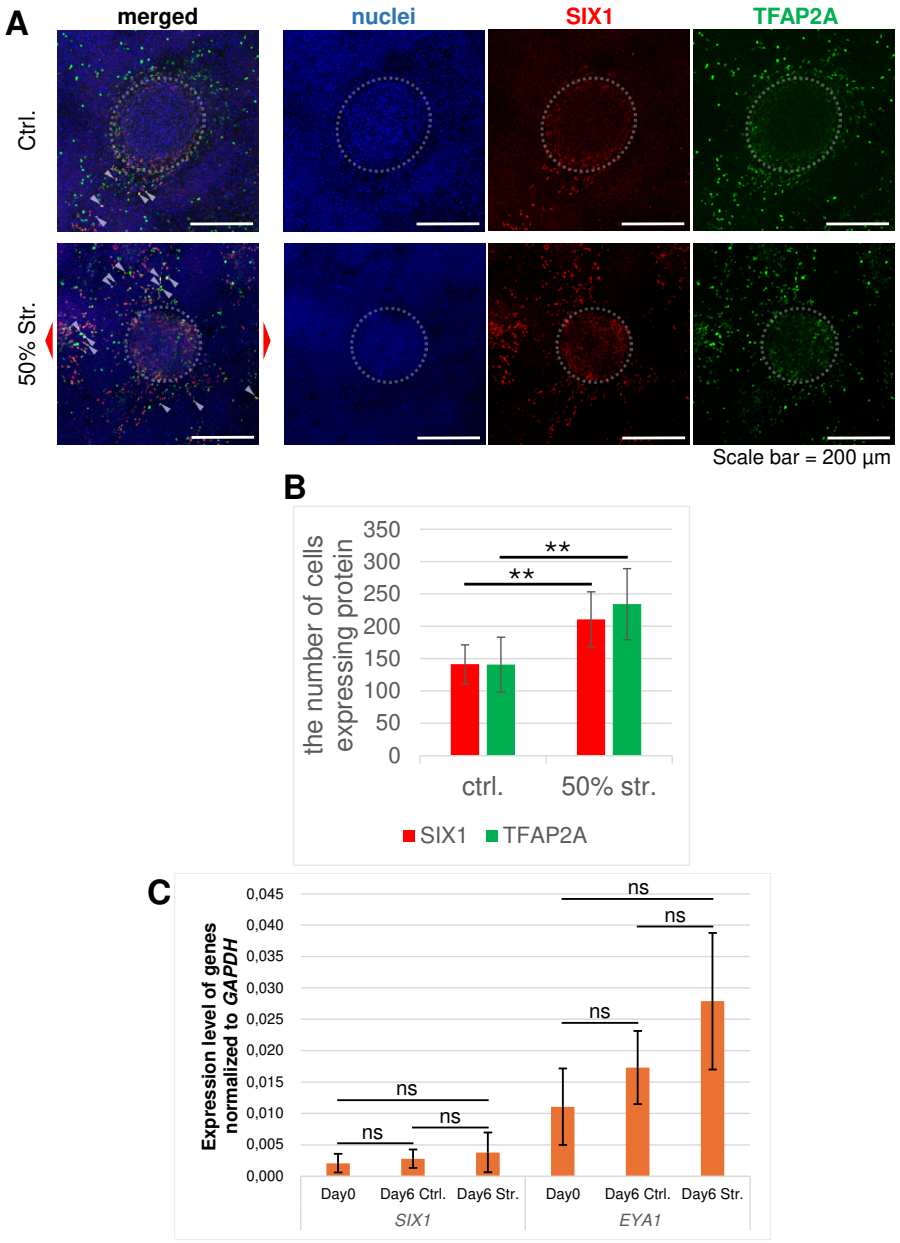
Supplementary Figure 2. Five-subdivided stacked slice along to Z-axis of Fig. 1E, F. (A) Stacked slices from the bottom to top of Fig. 1E. White arrow heads indicate TFAP2A in the hemispherical colony. The interval of each slice is 15 µm, Scale bar = 200 µm. **(B)** Stacked slices from the bottom to top of Fig. 1F. The interval of each slice is 10 µm, Scale bar = 100 µm.



Supplementary Figure 3. The features of automatic stretcher and their components. (A) The stand (metal parts) prevents the body of manual stretcher from being upside down by the torque when the chamber is stretched. The screw of manual stretcher was conjoined to the stepping motor by the joint, and the motors were controlled by Arduino. The speed of stretching and relaxation was approximately 1 cm per 75 s. Plastic covers were also used to prohibit contamination. **(B)** The circuit diagram of the automatic stretcher. It includes not only driver boards for driving the motors, but also real time clock and display modules for the timer. Each of the parts were connected to Arduino through a breadboard.



Supplementary Figure 4. Defining of stretch rate and quantification of actual extension of cells. (A) The feature of stretch chamber in the control (left) and stretched (right) state. l_{Ctrl} and l_{Str} is the width of centerline at the well parallel to the stretching direction. **(B)** Quantification of actual extension of cells in 50% stretching. Cell membranes were traced and approximated by ellipse fitting method. a_S, a_C are semi major axes in stretched and control state, respectively. **(C)** Statistical analysis of r_{SMA} at different points of one layer in 50% stretching. The box indicates the IQR, spanning from the 25th percentile to the 75th percentile. The line inside the box shows the average, whereas X shows the median. The averages of r_{SMA} in each point were 1.26 ± 0.16 for (1, 3'), 1.28 ± 0.21 for (4, 1'), and 1.41 ± 0.19 (3, 2'). *n* is the cell number, two sample *t*-test was performed.



Supplementary Figure 5. SIX1$^+$ cells were also detected in the cell sheet by stretching stimuli. (A) Separation of merged immunofluorescent images in Fig. 2C. Red arrowheads indicate the direction of stretching. White arrowheads indicate co-expression of SIX1 and TFAP2A. **(B)** Quantification of the cell number in (A). *n* = 6 (SIX1), 5 (TFAP2A); *t*-test. **(C)** Expression level of PPE markers normalized to *GAPDH* in 30% stretching stimuli and control samples at Day 6. Their expression level changes were not significant compared with undifferentiated cells (Day 0). *n* = 3; paired *t*-test.

```
#include <AccelStepper.h>
#include <Arduino.h>
#include <TM1637Display.h>
#include <DS3231M.h>

//4096->360 deg
const long steps = 4096;
//relay, signal pin
int relay1 = 50;
int relay2 = 51;
int relay3 = 52;
// 8 pin switch
int switch1pin = 11;
int switch2pin = 10;
int switch3pin = 9;
int switch4pin = 8;
int switch5pin = 7;
int switch6pin = 6;
int switch7pin = 5;
int switch8pin = 4;
bool switch1 = 0;
bool switch2 = 0;
bool switch3 = 0;
bool switch4 = 0;
bool switch5 = 0;
bool switch6 = 0;
bool switch7 = 0;
bool switch8 = 0;
bool buttonFlag = false;

/* IN1->pin8
* IN2->pin9
* IN3->pin10
* IN4->pin11
*/
AccelStepper myStepper1(AccelStepper::HALF4WIRE, 31, 35, 33, 37);
AccelStepper myStepper2(AccelStepper::HALF4WIRE, 23, 27, 25, 29);
AccelStepper myStepper3(AccelStepper::HALF4WIRE, 22, 26, 24, 28);
AccelStepper myStepper4(AccelStepper::HALF4WIRE, 30, 34, 32, 36);

void Step(long count){
  myStepper1.moveTo(steps * count);
  myStepper2.moveTo(steps * count * 0.72);
  myStepper3.moveTo(steps * count * 0.55);
  myStepper4.moveTo(steps * count * 0.38);
  Serial.println("check in");
  while (1){
    myStepper1.run();
    myStepper2.run();
    myStepper3.run();
    myStepper4.run();
    if (myStepper1.distanceToGo() == 0){
      break;
    }
  }
  Serial.println("check out");
  myStepper1.setCurrentPosition(0);
  myStepper2.setCurrentPosition(0);
  myStepper3.setCurrentPosition(0);
  myStepper4.setCurrentPosition(0);
}
```

```
//------------------------------------------------

//TM1637 setting
#define CLK 3
#define DIO 2

//TM1637 variables
unsigned long s;
uint8_t data[] = { 0x00, 0x00, 0x00, 0x00 };

//timer variables
int hour;
int minute;
int second;

//timer setting
#define HOUR 1
#define MINUTE 58
#define SECOND 45

TM1637Display display(CLK, DIO);

//------------------------------------------------

/*
SDA -> A4
SCL -> A5
*/
DS3231M_Class DS3231M;

//------------------------------------------------

void TimerInit(){
  hour = HOUR;
  minute = MINUTE;
  second = SECOND;
  data[0] = display.encodeDigit(hour / 10);
  data[1] = 0x80 + display.encodeDigit(hour % 10);
  data[2] = display.encodeDigit(minute / 10);
  data[3] = display.encodeDigit(minute % 10);

  //brightness setting, 1(min.)~7(max.)
  display.setBrightness(0x07);
  display.setSegments(data);
  s = millis();
}

void TimerDisplay(){
  //initialize display
  TimerInit();

  //measurement of time
  unsigned int start = millis();

  //loop
  while(1){
    static uint8_t secs;
    DateTime now = DS3231M.now();
    //break out function with turned off timer
    //when hour, minute, second = 0
    if (hour == 0 && minute == 0 && second == 0){
      display.setBrightness(0x07, false);
      display.setSegments(data);
      return;
    }
```

```
    //when time remains 15 seconds, turns on outer power by relay module
    if (hour == 0 && minute == 0 && second == 15){
      digitalWrite(relay1, HIGH);
      digitalWrite(relay2, HIGH);
      digitalWrite(relay3, HIGH);
    }

    //colon OFF
    if (millis() - s > 500){
      data[1] = display.encodeDigit(hour % 10);
    }

    //when second = 1
    if (secs != now.second()){
      secs = now.second();
      Serial.println(String(now.hour()) + ":" + String(now.minute()) + ":" +
String(now.second()));
      data[1] = 0x80 + display.encodeDigit(hour % 10);
      second--;
      s = millis();
    }

    switch1 = digitalRead(switch1pin);
    switch2 = digitalRead(switch2pin);
    switch3 = digitalRead(switch3pin);
    switch4 = digitalRead(switch4pin);
    switch5 = digitalRead(switch5pin);
    switch6 = digitalRead(switch6pin);
    switch7 = digitalRead(switch7pin);
    switch8 = digitalRead(switch8pin);

    // push switch 1 (minus 1 hour)
    if (switch2 == LOW) {
      if (hour > 0) {
        if (buttonFlag == false) {
          hour -= 1;
          buttonFlag = true;
        }
      }
    }
    // push switch 3 (minus 30 minutes)
    if (switch4 == LOW) {
      if (hour*60 + minute >= 30) {
        if (buttonFlag == false) {
          minute -= 30;
          if (minute < 0) {
            minute += 60;
            hour -= 1;
          }
          buttonFlag = true;
        }
      }
    }
    // push switch 5 (minus 10 minutes)
    if (switch6 == LOW) {
      if (hour*60 + minute >= 10) {
        if (buttonFlag == false) {
          minute -= 10;
          if (minute < 0) {
            minute += 60;
            hour -= 1;
          }
          buttonFlag = true;
        }
```

```
    }
  }
  // push switch 7 (minus 1 minutes)
  if (switch8 == LOW) {
    if (hour*60 + minute >= 1) {
      if (buttonFlag == false) {
        minute -= 1;
        if (minute < 0) {
          minute += 60;
          hour -= 1;
        }
        buttonFlag = true;
      }
    }
  }
  // push switch 2 (plus 1 hour)
  if (switch1 == LOW) {
    if (buttonFlag == false) {
      hour += 1;
      buttonFlag = true;
    }
  }
  // push switch 4 (plus 30 minutes)
  if (switch3 == LOW) {
    if (buttonFlag == false) {
      minute += 30;
      if (minute >= 60) {
        minute -= 60;
        hour += 1;
      }
      buttonFlag = true;
    }
  }
  // push switch 6 (plus 10 minute)
  if (switch5 == LOW) {
    if (buttonFlag == false) {
      minute += 10;
      if (minute >= 60) {
        minute -= 60;
        hour += 1;
      }
      buttonFlag = true;
    }
  }
  // push switch 8 (plus 1 minute)
  if (switch7 == LOW) {
    if (buttonFlag == false) {
      minute += 1;
      if (minute >= 60) {
        minute -= 60;
        hour += 1;
      }
      buttonFlag = true;
    }
  }

  // button enable
  if (buttonFlag) {
    data[0] = display.encodeDigit(hour / 10);
    data[1] = 0x80 + display.encodeDigit(hour % 10);
    data[2] = display.encodeDigit(minute / 10);
    data[3] = display.encodeDigit(minute % 10);
    if ( switch1 == HIGH && switch2 == HIGH && switch3 == HIGH && switch4 ==
HIGH
      && switch5 == HIGH && switch6 == HIGH && switch7 == HIGH && switch8 ==
```

```
HIGH) {
       buttonFlag = false;
      }
    }

    //when second < 1
    if (second < 0){
      second = 59;
      if (minute == 0){
        minute = 59;
        hour--;
      }else{
        minute--;
      }
      data[0] = display.encodeDigit(hour / 10);
      data[1] = 0x80 + display.encodeDigit(hour % 10);
      data[2] = display.encodeDigit(minute / 10);
      data[3] = display.encodeDigit(minute % 10);

      //finish measurement of time (Ctrl + Shift + M)
      Serial.println(millis() - start);
    }
    display.setSegments(data);
  }
}

//-------------------------------------------------

void setup() {
  //relay control (pin 30)
  pinMode(relay1, OUTPUT);
  pinMode(relay2, OUTPUT);
  pinMode(relay3, OUTPUT);
  digitalWrite(relay1, HIGH);
  digitalWrite(relay2, HIGH);
  digitalWrite(relay3, HIGH);
  delay(15000UL);
  //speed, acceleration setting
  myStepper1.setMaxSpeed(1000.0);
  myStepper1.setAcceleration(500.0);
  myStepper1.setSpeed(200);
  myStepper2.setMaxSpeed(1000.0);
  myStepper2.setAcceleration(500.0);
  myStepper2.setSpeed(200);
  myStepper3.setMaxSpeed(1000.0);
  myStepper3.setAcceleration(500.0);
  myStepper3.setSpeed(200);
  myStepper4.setMaxSpeed(1000.0);
  myStepper4.setAcceleration(500.0);
  myStepper4.setSpeed(200);
  //initial position setting
  myStepper1.setCurrentPosition(0);
  myStepper2.setCurrentPosition(0);
  myStepper3.setCurrentPosition(0);
  myStepper4.setCurrentPosition(0);
  // 8 pin pull-up mode
  pinMode(switch1pin, INPUT_PULLUP);
  pinMode(switch2pin, INPUT_PULLUP);
  pinMode(switch3pin, INPUT_PULLUP);
  pinMode(switch4pin, INPUT_PULLUP);
  pinMode(switch5pin, INPUT_PULLUP);
  pinMode(switch6pin, INPUT_PULLUP);
  pinMode(switch7pin, INPUT_PULLUP);
  pinMode(switch8pin, INPUT_PULLUP);
```