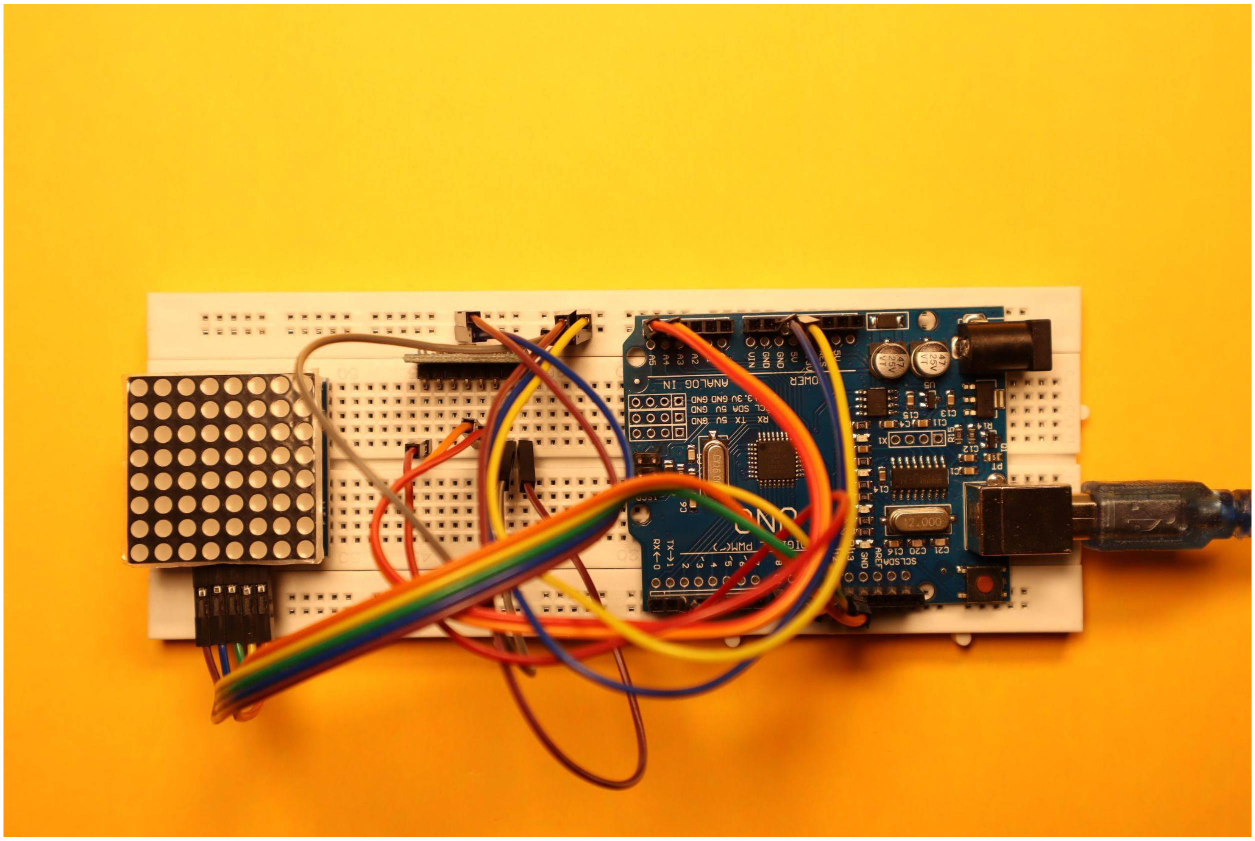
Robotix.io

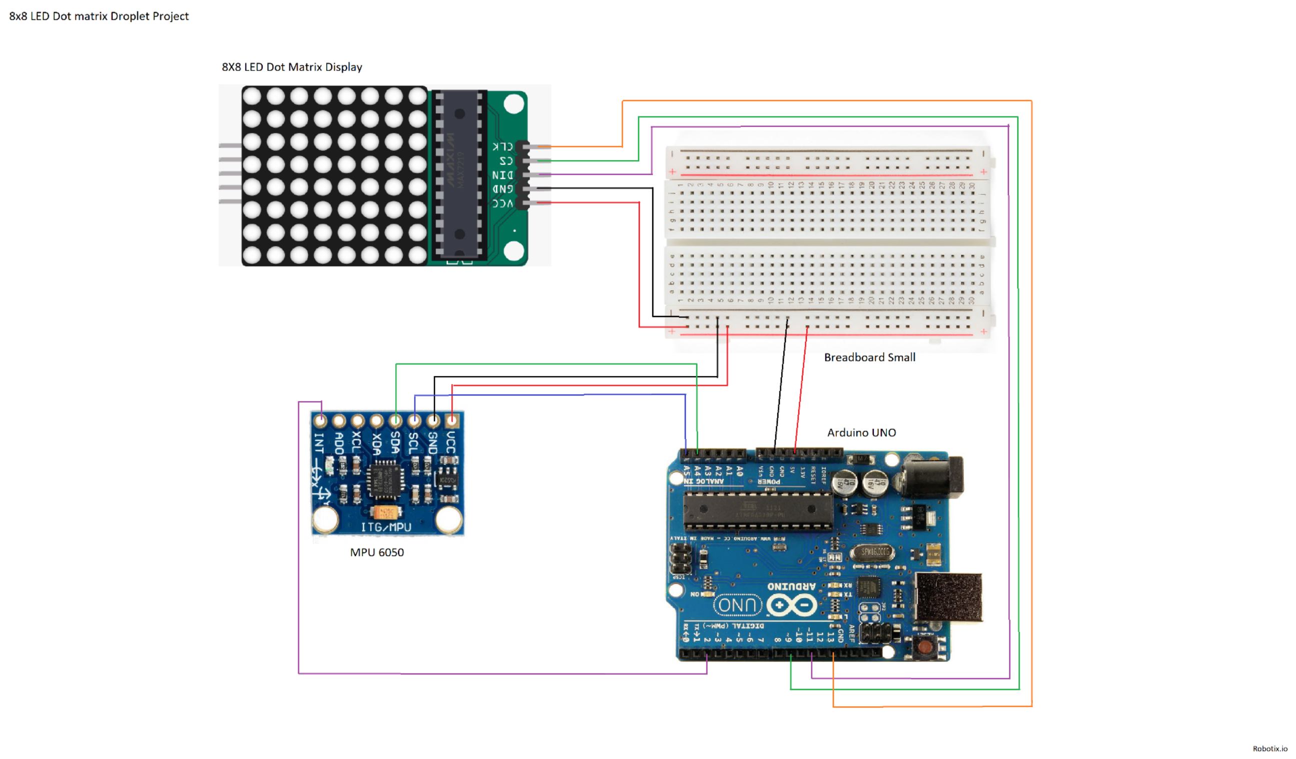
8X8 LED dot matrix droplet projects

Material Required 🧰 :

S No.	Components	Link
1	Arduino UNO	https://amzn.to/3R5IQPT
2	Breadboard Small / Large	https://amzn.to/3QdsROy
3	8X8 LED Dot matrix display	https://amzn.to/3RnVYPG
4	MPU6050	https://amzn.to/3Rz0lr5
5	Connecting Wires	https://amzn.to/3cI97EY
6	Arduino UNO Cable	https://amzn.to/3Cqxn8z



Circuit Diagram ⚡ :



Download These Libraries :

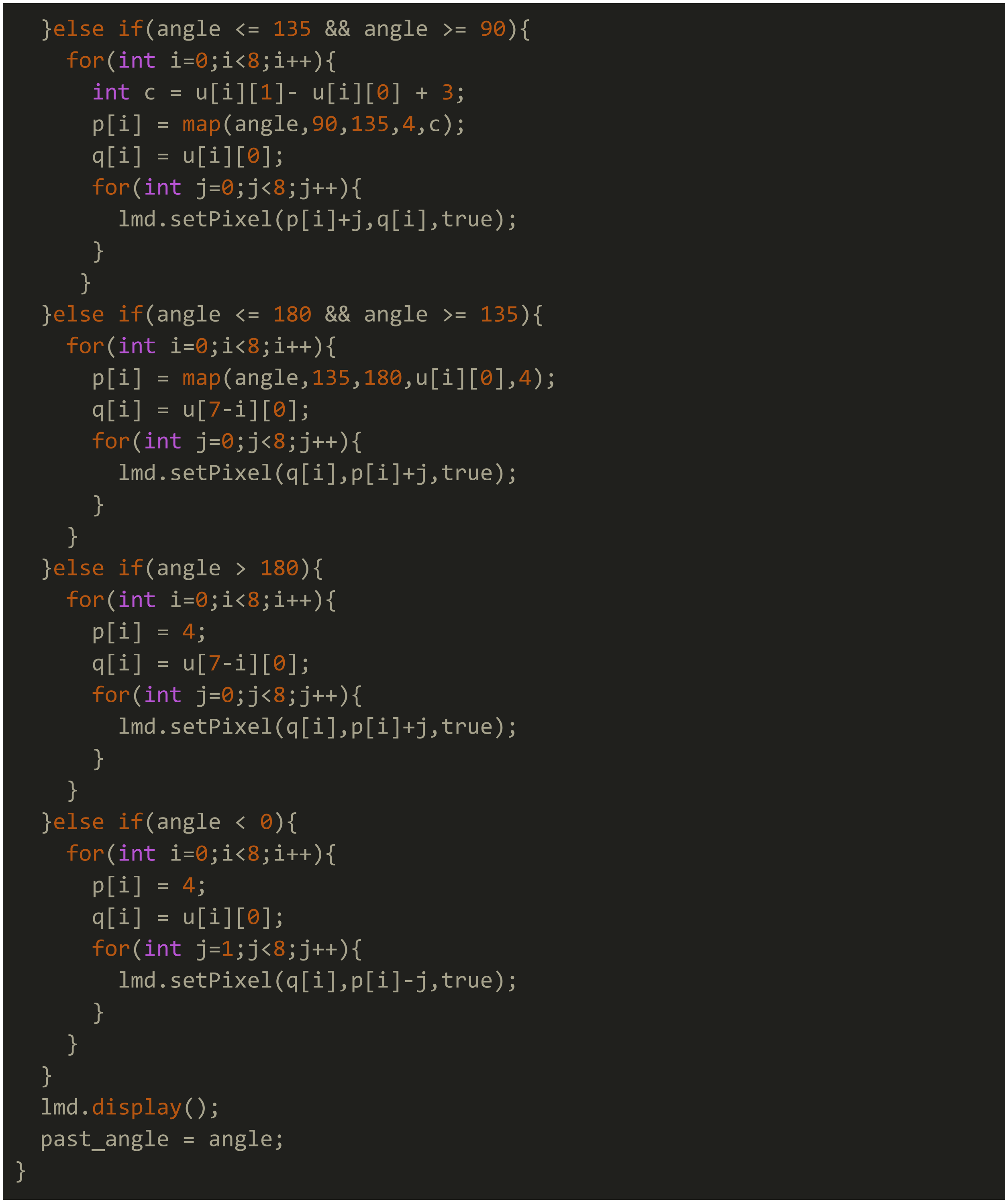
```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include <LEDMatrixDriver.hpp>
```

Code 💻 :

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include <LEDMatrixDriver.hpp>

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
```

```
#include "Wire.h"
#endif

//initiate mpu object
MPU6050 mpu;

const uint8_t LEDMATRIX_CS_PIN = 9;
const int LEDMATRIX_SEGMENTS = 1;
const int LEDMATRIX_WIDTH = LEDMATRIX_SEGMENTS * 8;
// The LEDMatrixDriver class instance
LEDMatrixDriver lmd(LEDMATRIX_SEGMENTS, LEDMATRIX_CS_PIN);

// MPU control/status vars
bool dmpReady = false; // set true if DMP init was successful
uint8_t mpuIntStatus; // holds actual interrupt status byte from MPU
uint8_t devStatus; // return status after each device operation (0 =
success, !0 = error)
uint16_t packetSize; // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount; // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q; // [w, x, y, z] quaternion container
VectorFloat gravity; // [x, y, z] gravity vector
float ypr[3]; // [yaw, pitch, roll] yaw/pitch/roll container and gravity
vector
float yaw, pitch, roll;
int count = 0;
volatile bool mpuInterrupt = false; // indicates whether MPU interrupt pin
has gone high

int past_angle;

void dmpDataReady()
{
  mpuInterrupt = true;
}

void setup()
{
// join I2C bus (I2Cdev library doesn't do this automatically)
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  Wire.begin();
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz)
#elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
  Fastwire::setup(400, true);
#endif

mpu.initialize();
Serial.begin(9600);
devStatus = mpu.dmpInitialize();


// supply your own gyro offsets here, scaled for min sensitivity
mpu.setXGyroOffset(220);
mpu.setYGyroOffset(76);
mpu.setZGyroOffset(-85);
mpu.setZAccelOffset(1788); // 1688 factory default for my test chip

// make sure it worked (returns 0 if so)
  if (devStatus == 0)
  {
    // turn on the DMP, now that it's ready
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    attachInterrupt(0, dmpDataReady, RISING);
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop() function knows it's okay
to use it
    dmpReady = true;

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();

  }
  else
  {
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed (code "));
    Serial.print(devStatus);
    Serial.println(F(")"));
  }

    // init the display
  lmd.setEnabled(true);
  lmd.setIntensity(2);   // 0 = low, 10 = hig
}


void loop()
{
  // if programming failed, don't try to do anything
  if (!dmpReady) return;

  // wait for MPU interrupt or extra packet(s) available
  while (!mpuInterrupt && fifoCount < packetSize);

  // reset interrupt flag and get INT_STATUS byte
  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();

  // get current FIFO count
  fifoCount = mpu.getFIFOCount();

  // check for overflow (this should never happen unless our code is too
inefficient)
  if ((mpuIntStatus & 0x10) || fifoCount == 1024)
  {
    // reset so we can continue cleanly
    mpu.resetFIFO();
    Serial.println(F("FIFO overflow!"));

  // otherwise, check for DMP data ready interrupt (this should happen
frequently)
  }
  else if (mpuIntStatus & 0x02)
  {
    // wait for correct available data length, should be a VERY short wait
    while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
    // read a packet from FIFO
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    // track FIFO count here in case there is > 1 packet available
    // (this lets us immediately read more without waiting for an
interrupt)
```

```
  fifoCount -= packetSize;

  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
  roll = (ypr[2] * 180/M_PI);


}
//array variables as dot containers
int u[8][2];
int v[8][2];
int m[8],n[8];
int p[8],q[8];
//fill initial dot sprite (horizontal line)
for(int i=0;i<8;i++){
    u[i][0] = i;
    u[i][1] = 4;
}
double angle = roll+90;
Serial.println(angle);
if(angle != past_angle){
  lmd.clear();
}

//algorithm for printing the dots according to the tilt angle
if(angle <= 90 && angle >= 45){
  for(int i=0;i<8;i++){
    int a = u[i][1]+ u[i][0] - 4;
    m[i] = map(angle,90,45,u[i][1],a);
    n[i] = u[i][0];
    for(int j=0;j<8;j++){
      lmd.setPixel(m[i]+j,n[i],true);
    }
  }
}else if(angle <= 45 && angle >= 0){
  for(int i=0;i<8;i++){
    p[i] = map(angle,45,0,u[i][0],4);
    q[i] = u[i][0];
    for(int j=1;j<8;j++){
      lmd.setPixel(q[i],p[i]-j,true);
    }
  }
```



```
  }else if(angle <= 135 && angle >= 90){
    for(int i=0;i<8;i++){
      int c = u[i][1]- u[i][0] + 3;
      p[i] = map(angle,90,135,4,c);
      q[i] = u[i][0];
      for(int j=0;j<8;j++){
        lmd.setPixel(p[i]+j,q[i],true);
      }
     }
  }else if(angle <= 180 && angle >= 135){
    for(int i=0;i<8;i++){
      p[i] = map(angle,135,180,u[i][0],4);
      q[i] = u[7-i][0];
      for(int j=0;j<8;j++){
        lmd.setPixel(q[i],p[i]+j,true);
      }
    }
  }else if(angle > 180){
    for(int i=0;i<8;i++){
      p[i] = 4;
      q[i] = u[7-i][0];
      for(int j=0;j<8;j++){
        lmd.setPixel(q[i],p[i]+j,true);
      }
    }
  }else if(angle < 0){
    for(int i=0;i<8;i++){
      p[i] = 4;
      q[i] = u[i][0];
      for(int j=1;j<8;j++){
        lmd.setPixel(q[i],p[i]-j,true);
      }
    }
  }
  lmd.display();
  past_angle = angle;
}
```