Embedded code:

/Smart watch arduino code:  
#include <NTPClient.h>  
#include <ESP8266WiFi.h>  
#include <WiFiUdp.h>  
#include <SPI.h>  
#include <Wire.h>  
#include <Adafruit\_GFX.h>  
#include <Adafruit\_SSD1306.h>  
#include <Adafruit\_Sensor.h>  
#include <Adafruit\_ADXL345\_U.h>  
  
  
#define SCREEN\_WIDTH 128 // OLED display width, in pixels  
#define SCREEN\_HEIGHT 32 // OLED display height, in pixelsdev  
  
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)  
// The pins for I2C are defined by the Wire-library.  
// On an arduino UNO:       A4(SDA), A5(SCL)  
// On an arduino MEGA 2560: 20(SDA), 21(SCL)  
// On an arduino LEONARDO:   2(SDA),  3(SCL), ...  
#define OLED\_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)  
#define SCREEN\_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32  
Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &Wire, OLED\_RESET);  
Adafruit\_ADXL345\_Unified accel = Adafruit\_ADXL345\_Unified(12345);  
const char \*ssid     = "IOTLAB";  
const char \*password = "Iotlab@123";  
  
const long utcOffsetInSeconds = 19800;  //UTC +5:30 E  For UTC -5.00 : -5 \* 60 \* 60 : -18000  
  
char daysOfTheWeek[7][12] = {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};  
  
// Define NTP Client to get time  
WiFiUDP ntpUDP;  
NTPClient timeClient(ntpUDP, "[pool.ntp.org](http://pool.ntp.org/)", utcOffsetInSeconds);  
//int LED\_BUILTIN = 14;  
  
  
  
void setup(){  
   
  Serial.begin(115200);  
   
   if(!display.begin(SSD1306\_SWITCHCAPVCC, SCREEN\_ADDRESS)) {  
    Serial.println(F("SSD1306 allocation failed"));  
    for(;;); // Don't proceed, loop forever  
  }  
    Serial.println("Accelerometer Test"); Serial.println("");  
   
  /\* Initialise the sensor \*/  
  if(!accel.begin())  
  {  
    /\* There was a problem detecting the ADXL345 ... check your connections \*/  
    Serial.println("Ooops, no ADXL345 detected ... Check your wiring!");  
    while(1);  
  }  
  
  /\* Set the range to whatever is appropriate for your project \*/  
  accel.setRange(ADXL345\_RANGE\_16\_G);  
  displaySensorDetails();  
   
  /\* Display additional settings (outside the scope of sensor\_t) \*/  
  displayDataRate();  
  displayRange();  
  Serial.println("");  
  
  pinMode(LED\_BUILTIN,OUTPUT);  
  // Clear the buffer  
  display.clearDisplay();  
  
  // Draw a single pixel in white  
  display.drawPixel(10, 10, SSD1306\_WHITE);  
  
  // Show the display buffer on the screen. You MUST call display() after  
  // drawing commands to make them visible on screen!  
  display.display();  
  delay(2000);  
  // display.display() is NOT necessary after every single drawing command,  
  // unless that's what you want...rather, you can batch up a bunch of  
  // drawing operations and then update the screen all at once by calling  
  // display.display(). These examples demonstrate both approaches...  
  
  WiFi.begin(ssid, password);  
  testdrawrect();  
  while ( WiFi.status() != WL\_CONNECTED ) {  
    delay ( 500 );  
    Serial.print ( "." );  
  }  
  testdrawroundrect();  
  timeClient.begin();  
 //digitalWrite(LED\_BUILTIN,HIGH);  
}  
  
void loop() {  
  /\* Get a new sensor event \*/  
  sensors\_event\_t event;  
  accel.getEvent(&event);  
   
  /\* Display the results (acceleration is measured in m/s^2) \*/  
 /\* Serial.print("X: "); Serial.print(event.acceleration.x); Serial.print("  ");  
  Serial.print("Y: "); Serial.print(event.acceleration.y); Serial.print("  ");  
  Serial.print("Z: "); Serial.print(event.acceleration.z); Serial.print("  ");Serial.println("m/s^2 ");\*/  
  delay(500);  
  if(event.acceleration.x > 0){  
    digitalWrite(LED\_BUILTIN,HIGH);  
    Serial.println("IN");  
   
    delay(50);  
  }  
   
  if(event.acceleration.x < 0){  
    digitalWrite(LED\_BUILTIN,LOW);  
    Serial.println("OUT");  
     
    delay(50);  
  }    
  timeClient.update();  
  
 /\*Serial.print(daysOfTheWeek[timeClient.getDay()]);  
  Serial.print(", ");  
  Serial.print(timeClient.getHours());  
  Serial.print(":");  
  Serial.print(timeClient.getMinutes());  
  Serial.print(":");  
  Serial.println(timeClient.getSeconds());  
  Serial.println(timeClient.getFormattedTime());\*/  
  testdrawstyles();  
  delay(1000);  
  
}  
  
void testdrawrect(void) {  
  display.clearDisplay();  
  
  for(int16\_t i=0; i<display.height()/2; i+=2) {  
    display.drawRect(i, i, display.width()-2\*i, display.height()-2\*i, SSD1306\_WHITE);  
    display.display(); // Update screen with each newly-drawn rectangle  
    delay(1);  
  }  
  
  delay(2000);  
}  
  
void testdrawroundrect(void) {  
  display.clearDisplay();  
  
  for(int16\_t i=0; i<display.height()/2-2; i+=2) {  
    display.drawRoundRect(i, i, display.width()-2\*i, display.height()-2\*i,  
      display.height()/4, SSD1306\_WHITE);  
    display.display();  
    delay(1);  
  }  
  
  delay(2000);  
}  
  
void testdrawstyles(void) {  
  display.clearDisplay();  
  
  display.setTextSize(1);             // Normal 1:1 pixel scale  
  display.setTextColor(SSD1306\_WHITE);     // Draw 'inverse' text    
  display.setCursor(5,0);             // Start at top-left corner  
  display.println(F("PBRVITS SMART WATCH"));  
  
  
  display.setTextSize(2);             // Draw 2X-scale text  
  display.setTextColor(SSD1306\_WHITE);  
  //display.print(F("0x"));  
  display.setCursor(40,10);  
  display.print(timeClient.getHours());  
  display.print(F(":"));  
  display.println(timeClient.getMinutes());  
   
  display.setTextSize(1);             // Normal 1:1 pixel scale  
  display.setTextColor(SSD1306\_WHITE);     // Draw 'inverse' text    
  display.setCursor(30,24);             // Start at top-left corner  
  display.println(daysOfTheWeek[timeClient.getDay()]);  
  
  display.display();  
  delay(2000);  
}  
void displaySensorDetails(void)  
{  
  sensor\_t sensor;  
  accel.getSensor(&sensor);  
  Serial.println("------------------------------------");  
  Serial.print  ("Sensor:       "); Serial.println([sensor.name](http://sensor.name/));  
  Serial.print  ("Driver Ver:   "); Serial.println(sensor.version);  
  Serial.print  ("Unique ID:    "); Serial.println(sensor.sensor\_id);  
  Serial.print  ("Max Value:    "); Serial.print(sensor.max\_value); Serial.println(" m/s^2");  
  Serial.print  ("Min Value:    "); Serial.print(sensor.min\_value); Serial.println(" m/s^2");  
  Serial.print  ("Resolution:   "); Serial.print(sensor.resolution); Serial.println(" m/s^2");    
  Serial.println("------------------------------------");  
  Serial.println("");  
  delay(500);  
}  
  
void displayDataRate(void)  
{  
  Serial.print  ("Data Rate:    ");  
   
  switch(accel.getDataRate())  
  {  
    case ADXL345\_DATARATE\_3200\_HZ:  
      Serial.print  ("3200 ");  
      break;  
    case ADXL345\_DATARATE\_1600\_HZ:  
      Serial.print  ("1600 ");  
      break;  
    case ADXL345\_DATARATE\_800\_HZ:  
      Serial.print  ("800 ");  
      break;  
    case ADXL345\_DATARATE\_400\_HZ:  
      Serial.print  ("400 ");  
      break;  
    case ADXL345\_DATARATE\_200\_HZ:  
      Serial.print  ("200 ");  
      break;  
    case ADXL345\_DATARATE\_100\_HZ:  
      Serial.print  ("100 ");  
      break;  
    case ADXL345\_DATARATE\_50\_HZ:  
      Serial.print  ("50 ");  
      break;  
    case ADXL345\_DATARATE\_25\_HZ:  
      Serial.print  ("25 ");  
      break;  
    case ADXL345\_DATARATE\_12\_5\_HZ:  
      Serial.print  ("12.5 ");  
      break;  
    case ADXL345\_DATARATE\_6\_25HZ:  
      Serial.print  ("6.25 ");  
      break;  
    case ADXL345\_DATARATE\_3\_13\_HZ:  
      Serial.print  ("3.13 ");  
      break;  
    case ADXL345\_DATARATE\_1\_56\_HZ:  
      Serial.print  ("1.56 ");  
      break;  
    case ADXL345\_DATARATE\_0\_78\_HZ:  
      Serial.print  ("0.78 ");  
      break;  
    case ADXL345\_DATARATE\_0\_39\_HZ:  
      Serial.print  ("0.39 ");  
      break;  
    case ADXL345\_DATARATE\_0\_20\_HZ:  
      Serial.print  ("0.20 ");  
      break;  
    case ADXL345\_DATARATE\_0\_10\_HZ:  
      Serial.print  ("0.10 ");  
      break;  
    default:  
      Serial.print  ("???? ");  
      break;  
  }    
  Serial.println(" Hz");    
}  
  
void displayRange(void)  
{  
  Serial.print  ("Range:         +/- ");  
   
  switch(accel.getRange())  
  {  
    case ADXL345\_RANGE\_16\_G:  
      Serial.print  ("16 ");  
      break;  
    case ADXL345\_RANGE\_8\_G:  
      Serial.print  ("8 ");  
      break;  
    case ADXL345\_RANGE\_4\_G:  
      Serial.print  ("4 ");  
      break;  
    case ADXL345\_RANGE\_2\_G:  
      Serial.print  ("2 ");  
      break;  
    default:  
      Serial.print  ("?? ");  
      break;  
  }    
  Serial.println(" g");    
}