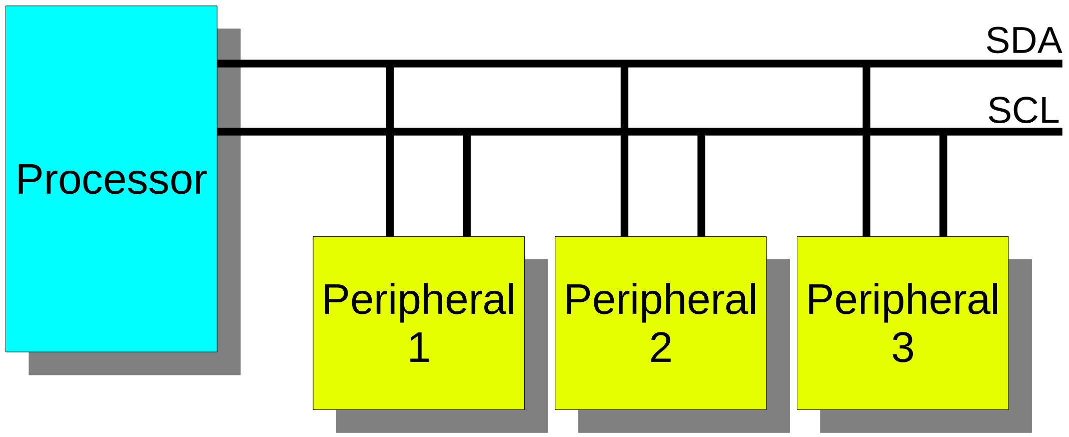
Proximity and Gesture Sensor and I2C Bus

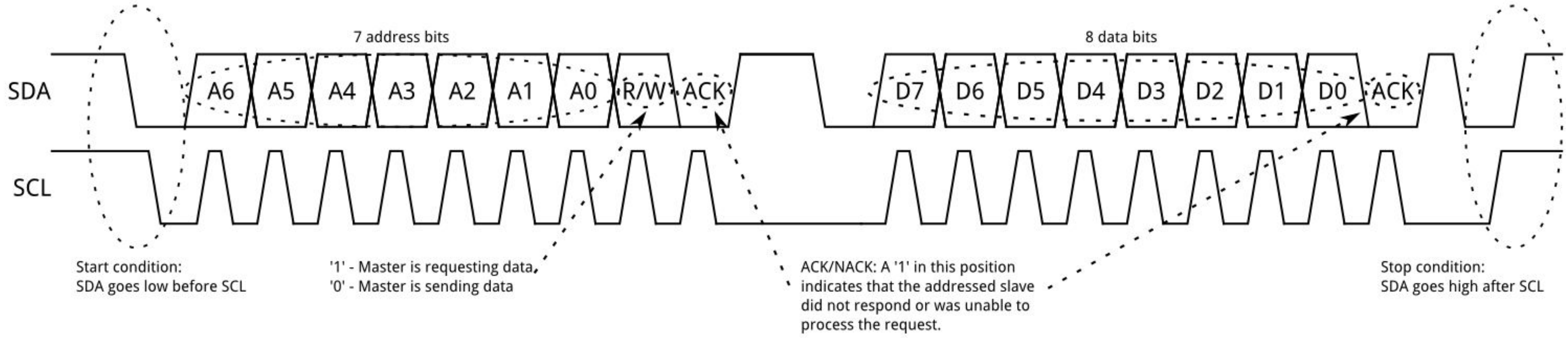
I2C Bus

- I2C stands for Inter-Integrated Circuit. It is a serial type interface requiring only two signals, a clock signal and a data signal.
- The I2C bus is typically used to interface with sensors and peripheral devices not needing to communicate at high speeds. The standard data rate is 100 Kilobits per second.
- Multiple devices can be connected to a single I2C bus. The processor is the controller and all the connected devices are peripherals. The processor is also called the master and the peripherals are slaves. Each slave has a unique address.



I2C Bus

- The clock signal is labeled SCL. This signal is used to control the flow of the data bits.
- The data signal is called SDA. This carries the data serially.
- The diagram below shows how a data transfer occurs.
- The data transfer protocol is for the master to first send out a device address. This is a 7 bit number followed by a bit indicating if the next byte is to be written to a slave or read from a slave.
- The SCL signal toggles for each bit sent.

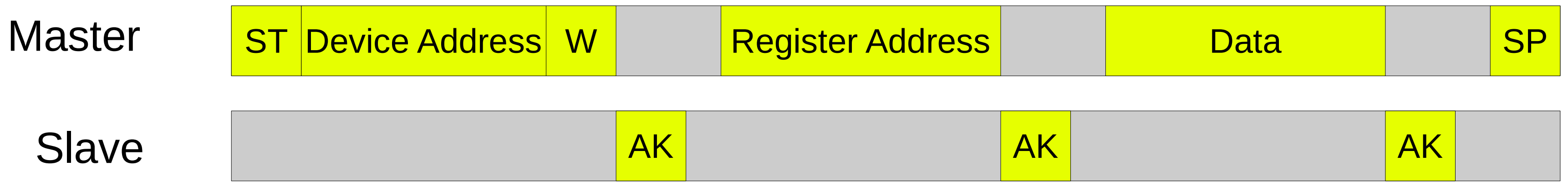

7 address bits
8 data bits
SDA
SCL
A6
A5
A4
A3
A2
A1
A0
R/W
ACK
D7
D6
D5
D4
D3
D2
D1
D0
ACK
Start condition:
SDA goes low before SCL
'1' - Master is requesting data
'0' - Master is sending data
ACK/NACK: A '1' in this position indicates that the addressed slave did not respond or was unable to process the request.
Stop condition:
SDA goes high after SCL

I2C Sequence

- Every device on the I2C bus has a unique 7-bit address. The gesture sensor address is 0x39.
- The I2C operation for writing to a register is:

  1. Send Start sequence by keeping SCL high and changing SDA from high to low (ST)
  2. Send the device address
  3. Send the register address
  4. Send the stop sequence by changing SDA from low to high while SCL is high first. (SP)

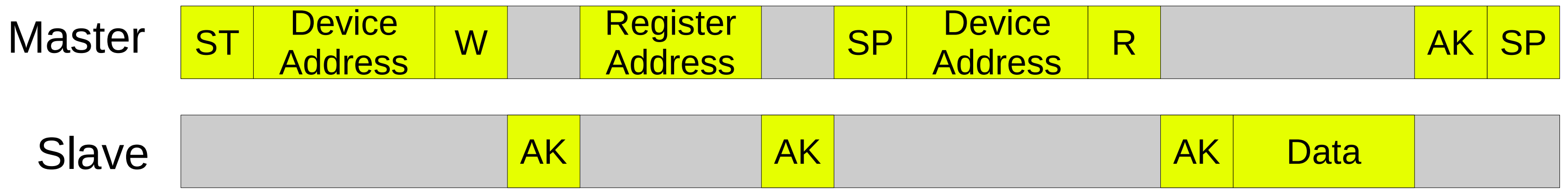
  - All Bytes sent are acknowledge by the slave.



Stensat Group LLC

I2C Sequence

- The I2C operation for reading a register is:
  1. Send Start sequence by keeping SCL high and changing SDA from high to low (ST)
  2. Send the device address
  3. Send the register address
  4. Send a repeat start sequence with a read indication (SR)
  5. Read in 6 bytes
  6. Send a stop sequence (SP)

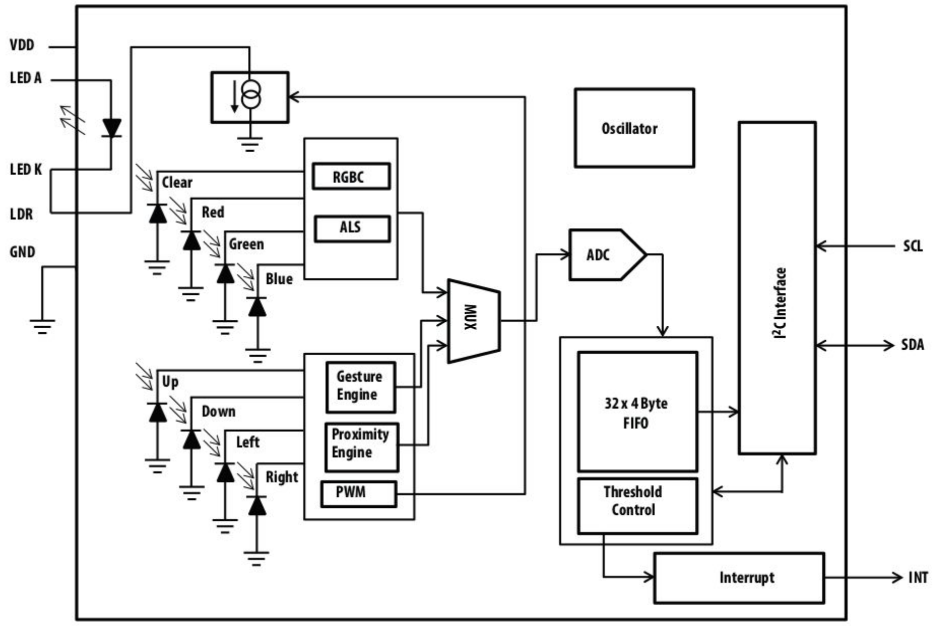


Gesture Sensor

- The APDS-9960 sensor can measure light level and the color levels in light. It can measure proximity of something in front of the sensor and it can also detect a set of hand gestures.
- Three programs will be shown. The programs are based on code from the Sparkfun library.

Gesture Sensor

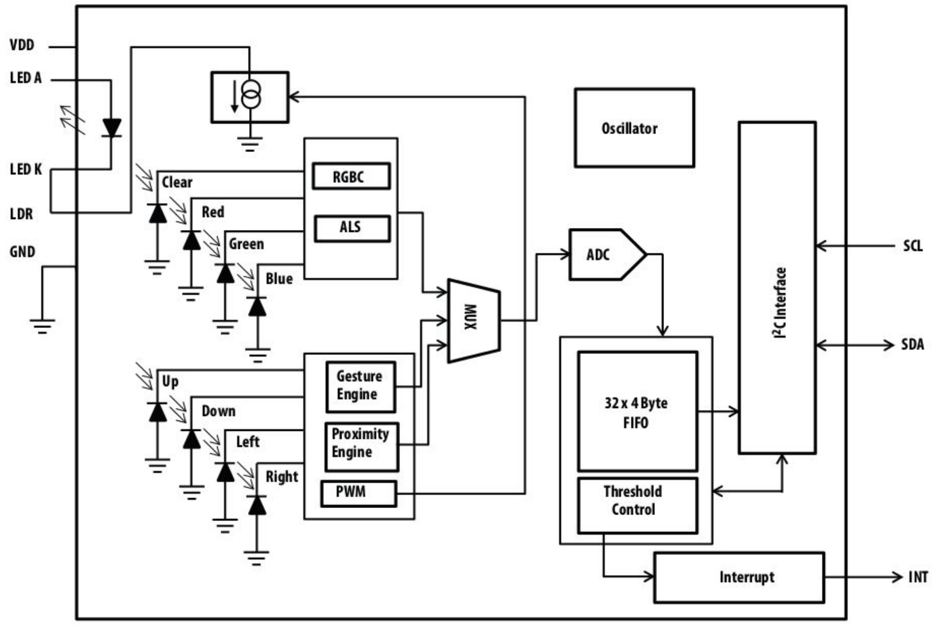
- The sensor includes one IR LED, four IR photodiodes for detecting gestures and four photodiodes for detecting ambient light level, red, green and blue.
- For proximity sensing, the IR LED is pulsed and the IR photodiodes detect the reflected light. The phasing of the pulsed light is used to determine distance. Range is not long, maybe 8 to 12 inches.
- For color detection, the four photodiodes with the clear, red, green and blue filters are used. Intensity of each is measured and made available.

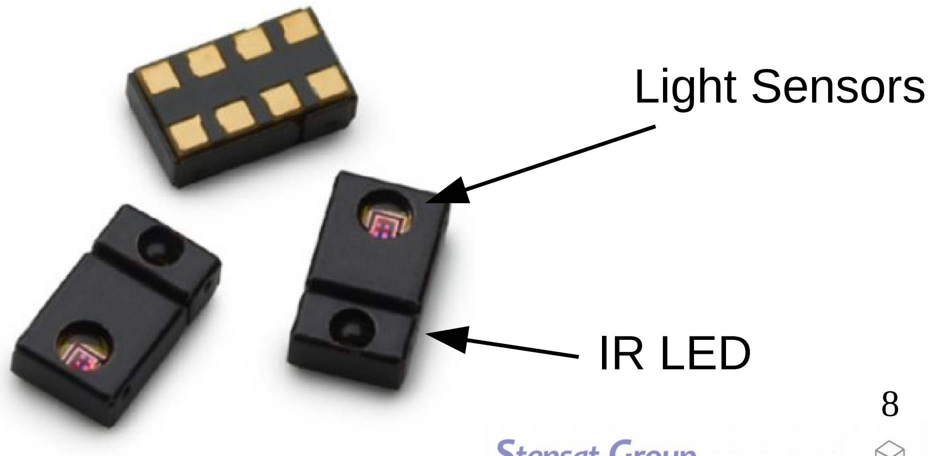


Stensat Group LLC

Gesture Sensor

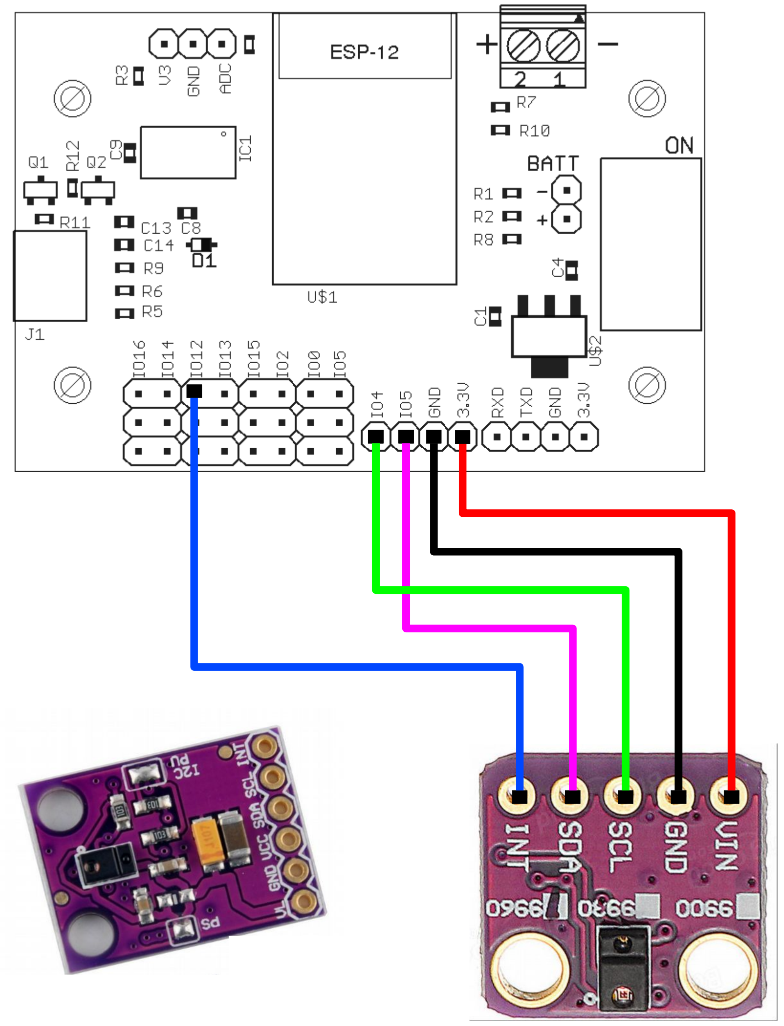
- For gesture sensing, the four IR photodiodes are are arranged in a diamond shape. They are used to detect changes in reflected IR light as a hand or object moves over head from one side to another. How the light changes and the timing the light changes is used to determine the gesture.





Connections

- The sensor needs to be connected to the I2C bus. The connections are shown in the picture.
  - VIN or VCC is connected to 3.3V.
  - SCL is connected to D5
  - SDA is connected to D4
  - GND is connected to GND.
  - INT is connected to D12. This is an interrupt signal that will be used in the gesture detection program.
- **There are two versions of this sensor. Both work the same. The pinouts are a bit different. Check the labeling on the sensor.**
- The sensor can be inserted into the solderless bread board and connected with male/female jumpers.



Stensat Group LLC

Install Library

- Before writing the program, the library supporting the gesture sensor needs to be installed.
- In the Arduino IDE, select **Sketch** menu and **Include Library**
- Select **Manage Libraries** at the top.
- Enter into the search at the top right, **APDS9960**.
- Select the **Sparkfun APDS9960 RGB and Gesture Sensor library**.
- Click Install.
- Once installed, close the Library Manager.

Proximity Sensor

- The sensor includes an IR emitter and detectors. The proximity sensor flashes the IR light and measures the time it takes for the light reflection to return.
- The Sparkfun library takes care of all the details.
- This code is from the examples and is stripped down to the bare minimum.

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

SparkFun_APDS9960 apds = SparkFun_APDS9960();
uint8_t proximity_data = 0;

void setup() {
  Wire.begin(4,5);
  Serial.begin(115200);
  if ( !apds.init() )
    Serial.println("Something went wrong");
  apds.setProximityGain(PGAIN_1X);
  }
  if (!apds.enableProximitySensor(false))
    Serial.println("Something went wrong);
}

void loop() {
  apds.readProximity(proximity_data) ) {
  Serial.print("Proximity: ");
  Serial.println(proximity_data);
  delay(250);
}
```

Proximity Sensor

- The Sparkfun library is included at the top of the program. An instance of the library device is created and named apds.
- In the **setup()** function, the I2C bus is configured along with the serial interface. The sensor is initialized with **apds.init()**. If it returns a 0, the if function will indicate something went wrong.
- The gain is set in **apds.setProximityGain()**.
- Last, the sensor is set to proximity mode with **apds.enableProximitySensor**.

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

SparkFun_APDS9960 apds = SparkFun_APDS9960();
uint8_t proximity_data = 0;

void setup() {
  Wire.begin(4,5);
  Serial.begin(115200);
  if ( !apds.init() )
    Serial.println("Something went wrong");
  apds.setProximityGain(PGAIN_1X);
  }
  if (!apds.enableProximitySensor(false))
    Serial.println("Something went wrong);
}

void loop() {
  apds.readProximity(proximity_data) ) {
  Serial.print("Proximity: ");
  Serial.println(proximity_data);
  delay(250);
}
```

Proximity Sensor

- In the loop() function, the proximity value is captured in the variable proximity_data.
- The function **apds.readProximity()** captures the measurement.
- Run the program and determine the measurement range.
- Change the gain and see if anything changes. PGAIN_1X can be changed to
  - PGAIN_2X
  - PGAIN_4X
  - PGAIN_8X

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

SparkFun_APDS9960 apds = SparkFun_APDS9960();
uint8_t proximity_data = 0;

void setup() {
  Wire.begin(4,5);
  Serial.begin(115200);
  if ( !apds.init() )
    Serial.println("Something went wrong");
  apds.setProximityGain(PGAIN_1X);
  }
  if (!apds.enableProximitySensor(false))
    Serial.println("Something went wrong);
}

void loop() {
  apds.readProximity(proximity_data) ) {
  Serial.print("Proximity: ");
  Serial.println(proximity_data);
  delay(250);
}
```

Color Sensing

- This example shows how to use the sensor to detect colors.
- Four variables are declared to hold the colors and intensity.
- In the **setup()** function, the sensor is initialized as before.
- The sensor is set to light sensor mode with **apds.enableLightSensor()**.
- A delay of a half second is inserted to allow the sensor to self calibrate.

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

SparkFun_APDS9960 apds = SparkFun_APDS9960();
uint16_t ambient_light = 0;
uint16_t red_light = 0;
uint16_t green_light = 0;
uint16_t blue_light = 0;

void setup() {
  Wire.begin(4,5);
  Serial.begin(115200);
  if (!apds.init() )
    Serial.println("Something went wrong");
  if (!apds.enableLightSensor(false) )
    Serial.println("Something went wrong");
  // Wait for calibration to finish
  delay(500);
}
```

Color Sensing

- In the loop function, the light intensity and four colors are captured separately. The big **if()** statement is used to detect any errors in reading any of the measurements. The symbol **||** means OR.
- If all data is collected, the results are displayed on the serial monitor.
- Light is sampled once a second.

```
void loop() {
  if (  !apds.readAmbientLight(ambient_light) ||
        !apds.readRedLight(red_light) ||
        !apds.readGreenLight(green_light) ||
        !apds.readBlueLight(blue_light) ) {
    Serial.println("Error reading light values");
  } else {
    Serial.print("Ambient: ");
    Serial.print(ambient_light);
    Serial.print(" Red: ");
    Serial.print(red_light);
    Serial.print(" Green: ");
    Serial.print(green_light);
    Serial.print(" Blue: ");
    Serial.println(blue_light);
  }
  delay(1000);
}
```

Gesture Sensing

- This example program uses the D12 digital port for generating an interrupt.
- An interrupt signal is a way to cause a function to be executed when an external event occurs. Digital port D12 is configured as an input and will generate an interrupt when the input signal transitions from high to low.

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

#define APDS9960_INT 12  //Interupt pin

SparkFun_APDS9960 apds = SparkFun_APDS9960();
volatile bool isr_flag = 0;

void interruptRoutine() {
  isr_flag = 1;
}

void setup() {
  Wire.begin(4,5);
  pinMode(12, INPUT);
  Serial.begin(115200);
  attachInterrupt(APDS9960_INT,
interruptRoutine, FALLING);
  if (!apds.init())
    Serial.println("Something went wrong");
  if (!apds.enableGestureSensor(true))
    Serial.println("Something went wrong
during gesture sensor init!");
}
```

Gesture Sensing

- The highlighted code shows the variable **isr_flag** will be used in the interrupt.
- **interruptRoutine()** will be executed every time an interrupt occurs. In the interrupt routine, the isr_flag variable is set to1. This is used to indicate the interrupt occurred. It is monitored in the loop() function.
- Notice the volatile word in the variable declaration. This is used to tell the compiler that the variable can be set anywhere at any time. This keeps the compiler from trying to optimize the code which could affect the variable.

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

#define APDS9960_INT 12  //Interupt pin

SparkFun_APDS9960 apds = SparkFun_APDS9960();
volatile bool isr_flag = 0;

void interruptRoutine() {
  isr_flag = 1;
}

void setup() {
  Wire.begin(4,5);
  pinMode(12, INPUT);
  Serial.begin(115200);
  attachInterrupt(APDS9960_INT, 
interruptRoutine, FALLING);
  if (!apds.init())
    Serial.println("Something went wrong");
  if (!apds.enableGestureSensor(true))
    Serial.println("Something went wrong 
during gesture sensor init!");
}
```

Gesture Sensing

- In **setup()** after initializing the I2C and serial interface, the interrupt is configured. Digital pin 12 which is named **APDS9660_INT** is set up as the trigger for the interrupt and that the trigger occurs only when the signal transitions from high to low. Any of the digital pins can be set up as an interrupt trigger. **RISING** is also valid for detecting when signals go from low to high.

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

#define APDS9960_INT 12  //Interupt pin

SparkFun_APDS9960 apds = SparkFun_APDS9960();
volatile bool isr_flag = 0;

void interruptRoutine() {
  isr_flag = 1;
}

void setup() {
  Wire.begin(4,5);
  pinMode(12, INPUT);
  Serial.begin(115200);
  attachInterrupt(APDS9960_INT,
interruptRoutine, FALLING);
  if (!apds.init())
    Serial.println("Something went wrong");
  if (!apds.enableGestureSensor(true))
    Serial.println("Something went wrong
during gesture sensor init!");
}
```

Gesture Sensing

- In **setup()** after initializing the I2C and serial interface, the interrupt is configured. Digital pin 12 which is named **APDS9660_INT** is set up as the trigger for the interrupt and that the trigger occurs only when the signal transitions from high to low. Any of the digital pins can be set up as an interrupt trigger. **RISING** is also valid for detecting when signals go from low to high.
- **apds.enableGestureSensor()** sets the sensor mode to detect gestures. The **true** argument enables the interrupt signal in the sensor.

```
#include <Wire.h>
#include <SparkFun_APDS9960.h>

#define APDS9960_INT 12  //Interupt pin

SparkFun_APDS9960 apds = SparkFun_APDS9960();
volatile bool isr_flag = 0;

void interruptRoutine() {
  isr_flag = 1;
}

void setup() {
  Wire.begin(4,5);
  pinMode(12, INPUT);
  Serial.begin(115200);
  attachInterrupt(APDS9960_INT,
interruptRoutine, FALLING);
  if (!apds.init())
    Serial.println("Something went wrong");
  if (!apds.enableGestureSensor(true))
    Serial.println("Something went wrong");
}
```

Gesture Sensing

- The loop function monitors the isr_flag and when the interrupt routine sets it to 1, the interrupt is disabled so no interrupts can occur while processing the current one. The function handleGesture() is called to process the sensor data. isr_flag is set to 0 and the interrupt is re-enabled for the next gesture.

```
void loop() {
  if( isr_flag == 1 ) {
    detachInterrupt(APDS9960_INT);
    handleGesture();
    isr_flag = 0;
    attachInterrupt(APDS9960_INT, interruptRoutine, FALLING);
  }
}
```

Gesture Sensing

- The **handleGesture()** function gets the gesture detected and displays the detected gesture.
- First, it checks to make sure a gesture result is available with **apds.isGestureAvailable()**.
- If so, the data is extracted from **apds.readGesture()**. The data is not assigned to a variable but used in the switch function directly.
- The values **DIR_UP** and others are defined in the library.

```
void handleGesture() {
    if ( apds.isGestureAvailable() ) {
    switch ( apds.readGesture() ) {
      case DIR_UP:
        Serial.println("UP");
        break;
      case DIR_DOWN:
        Serial.println("DOWN");
        break;
      case DIR_LEFT:
        Serial.println("LEFT");
        break;
      case DIR_RIGHT:
        Serial.println("RIGHT");
        break;
      case DIR_NEAR:
        Serial.println("NEAR");
        break;
      case DIR_FAR:
        Serial.println("FAR");
        break;
      default:
        Serial.println("NONE");
    }
  }
}
```