

Projet Prototypage Sanjay

Ryan Ossian

```
1 // Version 1.0.0
2
3 // Includes
4 #include <Arduino.h>
5 #include <Wire.h>
6 #include <SPI.h>
7 #include <EEPROM.h>
8
9 // Defines
10 #define BOUTON 2
11
12 // Variables
13 int etat_bouton = 0;
14
15 // Initialisation
16 void setup() {
17   pinMode(BOUTON, INPUT);
18   Serial.begin(9600);
19   Serial.println("Bouton pressé");
20   Serial.println("Bouton pressé");
21   Serial.println("Bouton pressé");
22 }
23
24 // Boucle principale
25 void loop() {
26   etat_bouton = digitalRead(BOUTON);
27   if (etat_bouton == 1) {
28     Serial.println("Bouton pressé");
29     delay(100);
30   }
31 }
```

```
1 // Version 1.0.0
2
3 // Includes
4 #include <Arduino.h>
5 #include <Wire.h>
6 #include <SPI.h>
7 #include <EEPROM.h>
8
9 // Defines
10 #define BOUTON 2
11
12 // Variables
13 int etat_bouton = 0;
14
15 // Initialisation
16 void setup() {
17   pinMode(BOUTON, INPUT);
18   Serial.begin(9600);
19   Serial.println("Bouton pressé");
20   Serial.println("Bouton pressé");
21   Serial.println("Bouton pressé");
22 }
23
24 // Boucle principale
25 void loop() {
26   etat_bouton = digitalRead(BOUTON);
27   if (etat_bouton == 1) {
28     Serial.println("Bouton pressé");
29     delay(100);
30   }
31 }
```

```
/*
 * Copyright (c) 2015 seeed technology inc.
 * Website : www.seeed.cc
 * Author : Wuruibin
 * Modified Time: June 2015
 * Description: This demo can recognize 9 gestures and output the result, including move up, move down, move
left, move right,
 * [ ] move forward, move backward, circle-clockwise, circle-counter clockwise, and wave.
 *
 * The MIT License (MIT)
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
```

- * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
- * copies of the Software, and to permit persons to whom the Software is
- * furnished to do so, subject to the following conditions:
- *
- * The above copyright notice and this permission notice shall be included in
- * all copies or substantial portions of the Software.
- *
- * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
- * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
- * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
- * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
- * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
- * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
- * THE SOFTWARE.
- */

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

```
#include "paj7620.h"
```

```
/*
```

Notice: When you want to recognize the Forward/Backward gestures, your gestures' reaction time must less than GES_ENTRY_TIME(0.8s).

You also can adjust the reaction time according to the actual circumstance.

```
*/
```

```
#define GES_REACTION_TIME 500 // You can adjust the reaction time according to the actual circumstance.
#define GES_ENTRY_TIME 800 // When you want to recognize the Forward/Backward gestures, your gestures
reaction time must less than GES_ENTRY_TIME(0.8s).
#define GES_QUIT_TIME 1000
```

```
void setup()
```

```
{
```

```
  uint8_t error = 0;
```

```
  Serial.begin(9600);
```

```
  Serial.println("\nPaj7620U2 TEST DEMO: Recognize 9 gestures.");
```

```
  error = paj7620Init(); // initialize Paj7620 registers
```

```
  if (error)
```

```
  {
```

```
    Serial.print("INIT ERROR, CODE:");
```

```

Serial.println(error);
}
else
{
Serial.println("INIT OK");
}
Serial.println("Please input your gestures:\n");
}

void loop()
{
uint8_t data = 0, data1 = 0, error;

error = paj7620ReadReg(0x43, 1, &data); // Read Bank_0_Reg_0x43/0x44 for gesture result.
if (!error)
{
switch (data) // When different gestures be detected, the variable 'data' will be set to different values b
paj7620ReadReg(0x43, 1, &data).
{
case GES_RIGHT_FLAG:
delay(GES_ENTRY_TIME);
paj7620ReadReg(0x43, 1, &data);
if(data == GES_FORWARD_FLAG)
{
Serial.println("Forward");
delay(GES_QUIT_TIME);
}
else if(data == GES_BACKWARD_FLAG)
{
Serial.println("Backward");
delay(GES_QUIT_TIME);
}
else
{
Serial.println("Right");
}
break;
case GES_LEFT_FLAG:
delay(GES_ENTRY_TIME);
paj7620ReadReg(0x43, 1, &data);

```

```

    if(data == GES_FORWARD_FLAG)
    {
        Serial.println("Forward");
        delay(GES_QUIT_TIME);
    }
    else if(data == GES_BACKWARD_FLAG)
    {
        Serial.println("Backward");
        delay(GES_QUIT_TIME);
    }
    else
    {
        Serial.println("Left");
    }
    break;
case GES_UP_FLAG:
    delay(GES_ENTRY_TIME);
    paj7620ReadReg(0x43, 1, &data);
    if(data == GES_FORWARD_FLAG)
    {
        Serial.println("Forward");
        delay(GES_QUIT_TIME);
    }
    else if(data == GES_BACKWARD_FLAG)
    {
        Serial.println("Backward");
        delay(GES_QUIT_TIME);
    }
    else
    {
        Serial.println("Up");
    }
    break;
case GES_DOWN_FLAG:
    delay(GES_ENTRY_TIME);
    paj7620ReadReg(0x43, 1, &data);
    if(data == GES_FORWARD_FLAG)
    {
        Serial.println("Forward");
        delay(GES_QUIT_TIME);
    }

```

```

    }
    else if(data == GES_BACKWARD_FLAG)
    {
        Serial.println("Backward");
        delay(GES_QUIT_TIME);
    }
    else
    {
        Serial.println("Down");
    }
    break;
case GES_FORWARD_FLAG:
    Serial.println("Forward");
    delay(GES_QUIT_TIME);
    break;
case GES_BACKWARD_FLAG:
    Serial.println("Backward");
    delay(GES_QUIT_TIME);
    break;
case GES_CLOCKWISE_FLAG:
    Serial.println("Clockwise");
    break;
case GES_COUNT_CLOCKWISE_FLAG:
    Serial.println("anti-clockwise");
    break;
default:
    paj7620ReadReg(0x44, 1, &data1);
    if (data1 == GES_WAVE_FLAG)
    {
        Serial.println("wave");
    }
    break;
}
}
delay(100);
}

```

```

#include <Wire.h>
#include "paj7620.h"

#define GES_REACTION_TIME 500
#define GES_ENTRY_TIME 800
#define GES_QUIT_TIME 1000

#define LED_PIN A2

void setup() {
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW); // Assurez-vous que la LED est éteinte au démarrage

    Serial.begin(9600);
    Serial.println("\nPAJ7620U2 TEST DEMO: Recognize gestures.");

    uint8_t error = paj7620Init();
    if (error) {
        Serial.print("INIT ERROR, CODE:");
        Serial.println(error);
    } else {
        Serial.println("INIT OK");
    }
    Serial.println("Please input your gestures:\n");
}

void loop() {
    uint8_t data = 0;
    uint8_t error = paj7620ReadReg(0x43, 1, &data);

    if (!error) {
        switch (data) {
            case GES_LEFT_FLAG:
                digitalWrite(LED_PIN, HIGH); // Allumer la LED
                Serial.println("LED ON - Left Gesture");
                delay(GES_QUIT_TIME);
                break;
            case GES_RIGHT_FLAG:
                digitalWrite(LED_PIN, LOW); // Éteindre la LED
                Serial.println("LED OFF - Right Gesture");
        }
    }
}

```

```

        delay(GES_QUIT_TIME);
        break;
    default:
        break;
    }
}
delay(100);
}

```

```

#include <Wire.h>
#include "paj7620.h"
#include <Servo.h>

#define GES_REACTION_TIME 500
#define GES_ENTRY_TIME 800
#define GES_QUIT_TIME 1000

#define LED_PIN A2
#define SERVO_PIN 9

Servo myservo;

void setup() {
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW); // Assurez-vous que la LED est éteinte au démarrage

    myservo.attach(SERVO_PIN);
    myservo.write(90); // Position neutre du servo

    Serial.begin(9600);
    Serial.println("\nPAJ7620U2 TEST DEMO: Recognize gestures.");

    uint8_t error = paj7620Init();
    if (error) {
        Serial.print("INIT ERROR, CODE:");
        Serial.println(error);
    } else {
        Serial.println("INIT OK");
    }

    Serial.println("Please input your gestures:\n");
}

```

```
}

void loop() {
    uint8_t data = 0;
    uint8_t error = paj7620ReadReg(0x43, 1, &data);

    if (!error) {
        switch (data) {
            case GES_LEFT_FLAG:
                digitalWrite(LED_PIN, HIGH); // Allumer la LED
                myservo.write(180); // Déplacer le servo à 180°
                Serial.println("LED ON - Servo 180 - Left Gesture");
                delay(500); // Laisser le temps au servo de bouger
                break;
            case GES_RIGHT_FLAG:
                digitalWrite(LED_PIN, LOW); // Éteindre la LED
                myservo.write(0); // Déplacer le servo à 0°
                Serial.println("LED OFF - Servo 0 - Right Gesture");
                delay(500); // Laisser le temps au servo de bouger
                break;
            default:
                break;
        }
    }
    delay(100);
}
```

Revision #1

Created 7 February 2025 13:42:09 by Apavou Alamellou Sanjay

Updated 7 February 2025 15:00:35 by Apavou Alamellou Sanjay