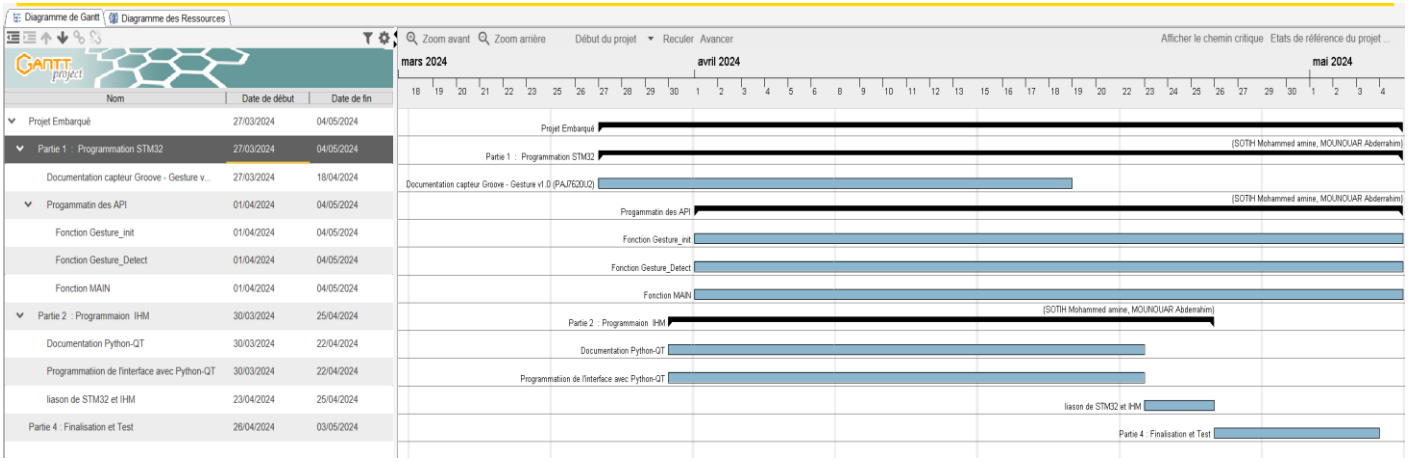
Guide de Prise en Main du Capteur de Gestes V1.0

CAPTEUR GESTES

V.1.0

Planification Gant du Projet



Guide de Prise en Main du Capteur de Gestes V1.0

Le capteur de gestes V1.0 est un dispositif conçu pour détecter et interpréter les mouvements de la main dans l'espace. Il permet d'ajouter des fonctionnalités interactives aux projets électroniques et informatiques. Ce guide vous expliquera comment configurer et utiliser le capteur de gestes V1.0.

Caractéristiques :

Détection de proximité intégrée : Le capteur peut détecter la présence d'objets à proximité.

Support de différentes cartes principales : Compatible avec Arduino UNO, Seeeduino, Arduino Mega2560, STM32.

9 Gestes de base :

- Haut
- Bas
- Gauche
- Droite
- Avant
- Arrière
- Sens horaire (Clockwise)
- Sens antihoraire (Counter Clockwise)
- Vague (Wave)

Matériel Requis

- Une carte microcontrôleur (par exemple, Arduino UNO, Seeeduino, Arduino Mega2560) dans notre cas STM32
- Câbles de connexion (si non fournis)
- Un ordinateur avec un logiciel de programmation (par exemple, Arduino IDE)
- Éditeur de texte(Vscode)
- Débogueur (DBG)
- OpenOCD
- Terminal (GTKTerm ou Minicom)

Utilisation du Débogueur et du Terminal :

Pour un développement et un débogage plus avancé, vous pouvez utiliser un débogueur (DBG) et OpenOCD. Voici les étapes générales :

Configuration du Débogueur :

1- Installez OpenOCD :

```
> sudo apt-get install openocd
```

2 - Lancez OpenOCD **:**

```
> openocd -f interface/your-debugger.cfg -f target/your-target.cfg
```

Utilisation de GTKTerm ou Minicom :

Pour surveiller la sortie série :

1 - Installez GTKTerm ou Minicom :

```
> sudo apt-get install gtkterm sudo
> apt-get install minicom
```

2 - Lancez le terminal :

- Pour GTKTerm :

```
> gtkterm
```

- Pour Minicom :

```
> minicom -b 9600 -o -D /dev/ttyUSB0
```

Assurez-vous de remplacer /dev/ttyUSB0 par le port série correct.

Environnement de développement :

Pour utiliser le capteur de gestes avec STM32, vous devez installer la bibliothèque appropriée :

Vous pouvez avoir le projet depuis gitlab : https://git.enib.fr/a3mounou/projet-capteur-de-gesture.git

Utilisation et Tests

- Branchez la carte STM32 à votre ordinateur (commande ocd & puis connecter).
- Chargez le programme sur votre STM32 en utilisant le Débogueur (DBG) **'dbg main.elf '**. (Attention n'oubliez pas de compiler le projet par un make dans le répertoire de votre projet)
- Ouvrez le Moniteur Série pour voir les résultats des gestes détectés.
- Testez différentes gestuelles (haut, bas, gauche, droite, avant, arrière, sens horaire, sens antihoraire, vague) pour vérifier la détection.

Code du capteur :

- La fonction **gesture_init()** pour initialiser notre capteur :

```
// Initialise les capteurs
void gesture_init() {
    // Initialiser l'interface I2C
    i2c_master_init(I2C1); // En supposant que vous utilisez I2C1
    gestureSelectBank(BANK0);

    // Vérifier l'ID du périphérique
    uint8_t data;
    data = gestureReadReg(0,1);
    //uart_printf(_USART2,"\r\ndata = %d",data);
    if (data != 0x20) {
        // Erreur avec le capteur
        uart_puts(_USART2, "\n\r Erreur avec le capteur");
        // cls();
        //  lcd_printf("*********ERREUR CAPTEUR*********");
        // leds(LED_RED );
        return;
    }
    if (data == 0x20)
            uart_printf(_USART2,"\r\nCapteur ON ");
           // cls();
           //  lcd_printf("**********Capteur ON*********");
           //leds(LED_GREEN);

    // Initialiser les registres
    for (int i = 0; i < sizeof(initRegisterArray) / sizeof(initRegisterArray[0]); i++) {
        gestureWriteReg(initRegisterArray[i][0], initRegisterArray[i][1]);
    }

    // Sélectionner la banque 0
    gestureSelectBank(BANK0);
    uart_printf(_USART2,"\r\nCapteur initialiser.");

    return;
}
```

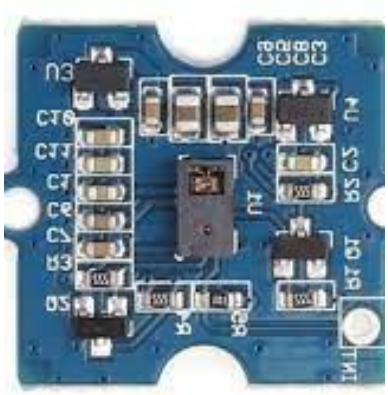
La fonction **print_gesture()** pour lire le geste effectuer et puis l'afficher :

```
void print_gesture() {
    uint8_t data = gestureReadReg(0x43, 1);
    //term_printf("je suis ici");

    if (data == GES_RIGHT_FLAG) {
        delay_us(GES_ENTRY_TIME);
        data = gestureReadReg(0x43, 1);
        if (data == GES_FORWARD_FLAG) {
            uart_printf(_USART2, "Forward \r\n");
            //zigbee_printf(_USART1, "Forward \r\n");
            delay_us(GES_QUIT_TIME);
        } else if (data == GES_BACKWARD_FLAG) {
            uart_printf(_USART2, "Backward\r\n");
            //zigbee_printf(_USART1, "Backward\r\n");
            delay_us(GES_QUIT_TIME);
        } else {
            uart_printf(_USART2, "Right\r\n");
            //zigbee_printf(_USART1, "Right\r\n");
        }
    } else if (data == GES_LEFT_FLAG) {
        delay_us(GES_ENTRY_TIME);
        data = gestureReadReg(0x43, 1);
        if (data == GES_FORWARD_FLAG) {
            uart_printf(_USART2, "Forward\n");
            //zigbee_printf(_USART1, "Forward\n");
            delay_us(GES_QUIT_TIME);
        } else if (data == GES_BACKWARD_FLAG) {
            uart_printf(_USART2, "Backward\r\n");
        }
    } else if (data == GES_LEFT_FLAG) {
        delay_us(GES_ENTRY_TIME);
        data = gestureReadReg(0x43, 1);
        if (data == GES_FORWARD_FLAG) {
            uart_printf(_USART2, "Forward\n");
            //zigbee_printf(_USART1, "Forward\n");
            delay_us(GES_QUIT_TIME);
        } else if (data == GES_BACKWARD_FLAG) {
            uart_printf(_USART2, "Backward\r\n");
           // zigbee_printf(_USART1, "Backward\r\n");
            delay_us(GES_QUIT_TIME);
        } else {
            uart_printf(_USART2, "Left\r\n");
            //zigbee_printf(_USART1, "Left\r\n");
        }
    } else if (data == GES_UP_FLAG) {
        delay_us(GES_ENTRY_TIME);
        data = gestureReadReg(0x43, 1);
        if (data == GES_FORWARD_FLAG) {
            uart_printf(_USART2, "Forward\r\n");
            //zigbee_printf(_USART1, "Forward\r\n");
            delay_us(GES_QUIT_TIME);
        } else if (data == GES_BACKWARD_FLAG) {
            uart_printf(_USART2, "Backward\r\n");
            //zigbee_printf(_USART1, "Backward\r\n");
            delay_us(GES_QUIT_TIME);
        } else {
            uart_printf(_USART2, "Up\r\n");
            //zigbee_printf(_USART1, "Up\r\n");
        }
    } else if (data == GES_DOWN_FLAG) {
        delay_us(GES_ENTRY_TIME);
        data = gestureReadReg(0x43, 1);
        if (data == GES_FORWARD_FLAG) {
            uart_printf(_USART2, "Forward\r\n");
            //zigbee_printf(_USART1, "Forward\r\n");
            delay_us(GES_QUIT_TIME);
        } else if (data == GES_BACKWARD_FLAG) {
            uart_printf(_USART2, "Backward\r\n");
            //zigbee_printf(_USART1, "Backward\r\n");
            delay_us(GES_QUIT_TIME);
        } else {
            uart_printf(_USART2, "Down\r\n");
            //zigbee_printf(_USART1, "Down\r\n");
        }
    } else if (data == GES_FORWARD_FLAG) {
        uart_printf(_USART2, "Forward\r\n");
        //zigbee_printf(_USART1, "Forward\r\n");
        delay_us(GES_QUIT_TIME);
    } else if (data == GES_BACKWARD_FLAG) {
        uart_printf(_USART2, "Backward\r\n");
        //zigbee_printf(_USART1, "Backward\r\n");
        delay_us(GES_QUIT_TIME);
    } else if (data == GES_CLOCKWISE_FLAG) {
        uart_printf(_USART2, "Clockwise\r\n");
        //zigbee_printf(_USART1, "Clockwise\r\n");
    } else if (data == GES_COUNT_CLOCKWISE_FLAG) {
        uart_printf(_USART2, "anti-clockwise\r\n");
        //zigbee_printf(_USART1, "anti-clockwise\r\n");
    } else {
        data = gestureReadReg(0x44, 1);
        if (data == GES_WAVE_FLAG) {
            uart_printf(_USART2, "wave\r\n");
            //zigbee_printf(_USART2, "wave\r\n");
        }
    }
}
```

Le capteur de gestes V1.0 est un outil puissant pour ajouter des interactions naturelles à vos projets. Avec une configuration appropriée et une calibration soignée, il peut fournir des résultats fiables et précis. Suivez ce guide pour débuter et expérimentez avec différentes applications de gestes.

Pour toute question supplémentaire ou pour des exemples de projets avancés, consultez la documentation en ligne fournie par le fabricant du capteur.



Groove gesture V1.0 :

Pour toute question supplémentaire ou pour des exemples de projets avancés, consultez la documentation en ligne fournie par le fabricant du capteur. Pour en savoir plus sur le capteur et ses fonctionnalités, veuillez visiter le site officiel de Grove https://wiki.seeedstudio.com/Grove-Gesture_v1.0/.