Rapport Technique - Station Météorologique
Projet Systèmes Embarqués
STM32F746NG Discovery

DASSI DASSI Samuel, BOUNGAB Mohammed, MEJNOUN Ibrahim

Décembre 2025

Table des matières

1 Introduction

Dans le cadre de ce projet, nous avons développé une station météorologique autonome capable de mesurer et d'analyser diverses variables climatiques en temps réel. Le système est basé sur la carte STM32F746NG Discovery et intègre plusieurs capteurs pour la collecte de données météorologiques précises.

L'objectif principal de ce projet est de concevoir une solution fiable et performante pour :

— Collecter des données météorologiques en temps réel (température, humidité, pression, vitesse et direction du vent, précipitations)
— Stocker ces données dans une carte SD au format CSV
— Visualiser les données via une interface homme-machine (IHM) tactile
— Gérer l'énergie du système avec des modes de veille automatiques

Le projet s'articule autour d'une architecture événementielle basée sur des interruptions, garantissant un fonctionnement non-bloquant et efficace. Chaque composant a été développé de manière modulaire, permettant une validation unitaire avant l'intégration finale.

2 Spécifications externes de l'application

2.1 Présentation des capteurs

2.1.1 Capteur de température et d'humidité (HTS221)

Le capteur HTS221 permet de mesurer :
— Humidité relative : 0 à 100%
— Température : -40°C à +120°C
— Communication : Interface I2C (bus I2C1, pins PB8/PB9)

2.1.2 Capteur de pression (LPS22HH)

Le capteur LPS22HH permet de mesurer :
— Pression atmosphérique : 260 hPa à 1260 hPa
— Communication : Interface I2C (bus I2C1)
— Précision adaptée aux conditions atmosphériques normales (environ 1013 hPa)

2.1.3 Capteur de vitesse du vent (Anémomètre)

Le capteur de vitesse du vent est composé d'une hélice avec des demi-sphères. Lorsque l'hélice effectue un tour complet, un aimant génère une impulsion détectée par le microcontrôleur.

— Mesure : Input Capture sur TIM1, Channel 1
— Conversion : Ticks par seconde convertis en km/h selon la formule : $v = \frac{ticks/s}{5} \times 2.4$
— Précision : Une impulsion correspond à un tour complet de l'hélice

2.1.4 Capteur de direction du vent (Girouette)

La girouette utilise un potentiomètre rotatif qui génère une tension variable selon l'orientation.

— Mesure : ADC2, Channel 1
— Tension de référence : 3.3V
— Résolution : 12 bits (4096 niveaux)
— Conversion : Table de correspondance tension/direction (16 directions)

2.1.5 Capteur de précipitations (Pluviomètre)

Le pluviomètre utilise un système mécanique qui génère une impulsion à chaque basculement.

— Détection : Interruption EXTI sur GPIO_PIN_15
— Débouncing : Logiciel avec vérification d'un délai minimum de 1 seconde
— Conversion : $Pluie(mm) = interrupt\_count \times 0.2794$

2.2 Principales fonctionnalités

Le système implémente les fonctionnalités suivantes :

1. **Acquisition périodique des données** : Mesure automatique des capteurs selon une période configurable (5 secondes, 10 minutes, 1 heure)
2. **Affichage en temps réel** : Visualisation des données sur l'écran LCD tactile (480x272 pixels)
3. **Stockage sur carte SD** : Sauvegarde périodique des données au format CSV avec horodatage RTC
4. **Interface tactile** : Navigation entre les écrans via le touchscreen
5. **Gestion de l'énergie** : Mise en veille automatique de l'écran après 60 secondes d'inactivité
6. **Indicateurs visuels** : LEDs pour indiquer l'état du système (vert = acquisition, rouge = veille)

2.3 Bords du modèle

Le diagramme suivant présente l'architecture matérielle complète du système, montrant les connexions entre le microcontrôleur STM32F746 et tous les périphériques externes (capteurs, écran, carte SD, etc.).

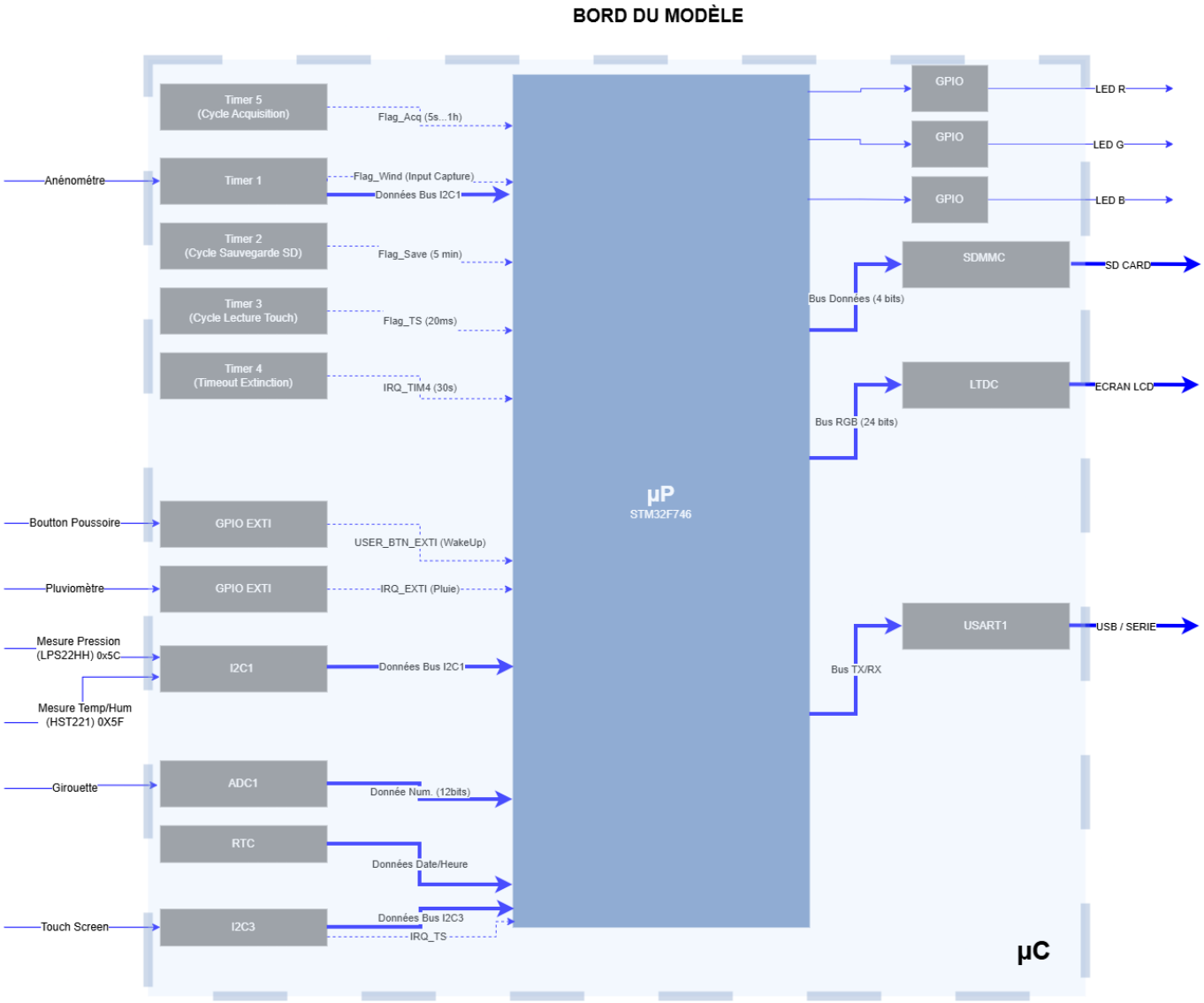


FIGURE 1 – Diagramme des bords du modèle - Architecture matérielle du système

Ce diagramme illustre :

— Les connexions I2C pour les capteurs HTS221 (0x5F) et LPS22HH (0x5C)
— Les timers utilisés pour la gestion des événements (TIM1, TIM2, TIM3, TIM4, TIM5)
— Les interruptions EXTI pour le pluviomètre et le bouton utilisateur
— Les interfaces de sortie : LCD (LTDC), carte SD (SDMMC1), LEDs (GPIO), UART (USART1)
— L'ADC pour la girouette et le touchscreen via I2C3

2.4 Tables des Entrées/Sorties

2.4.1 Table des entrées

TABLE 1 – Table des entrées du système

Signal	Type	Périphérique	Description
HTS221 Data	I2C1	PB8/PB9	Température et humidité
LPS22HH Data	I2C1	PB8/PB9	Pression atmosphérique
Anémomètre	TIM1 IC	PA8	Vitesse du vent
Girouette	ADC2 CH1	PA1	Direction du vent
Pluviomètre	EXTI15	PC15	Précipitations
Touchscreen	I2C3	PH7/PH8	Interface tactile
Bouton User	EXTI0	PA0	Réveil système

2.4.2 Table des sorties

TABLE 2 – Table des sorties du système

Signal	Type	Périphérique	Description
LCD Display	LTDC	Multiple	Affichage 480x272
LED Verte	GPIO	PJ2	État acquisition
LED Rouge	GPIO	PJ13	État veille
UART Debug	USART1	PA9/PA10	Console série
SD Card	SDMMC1	Multiple	Stockage données

2.5 Table des interruptions avec gestion des priorités

TABLE 3 – Table de priorité des interruptions (DFE)

Élément (DFE)	Type d'IRQ	IRQ NVIC (typique)	Priorité (préemption)	Pourquoi
TIM1 – Input Capture (front montant)	TIM Capture/-Compare	TIM1_CC_IRQn	0	Critique : vitesse vent → ne jamais rater les impulsions
Pluie (auget) – EXTI (front montant)	GPIO EXTI	EXTI..._IRQn	0	Critique : comptage pluie → ne jamais perdre un "tick"
TIM3 – Update (20 ms)	TIM Update	TIM3_IRQn	1	UI / tactile réactif
Bouton utilisateur – EXTI (appui)	GPIO EXTI	EXTI..._IRQn	2	Commande utilisateur (start/stop, mode, wakeup)
TIM5 – Update (toutes les 5 s)	TIM Update	TIM5_IRQn	3	Cycle acquisition périodique (jitter toléré)
TIM4 – Update (30 s)	TIM Update	TIM4_IRQn	4	Timeout écran : faible priorité
TIM2 – Update (toutes les 5 min)	TIM Update	TIM2_IRQn	5	Sauvegarde SD : lourd → IRQ basse + flag only

Note : La priorité la plus forte ici est 0 tandis que la plus faible est 5. Les interruptions avec priorité 0 (TIM1 Input Capture et Pluviomètre EXTI) sont critiques car elles ne doivent jamais perdre d'événements. Les interruptions avec priorité élevée (TIM2, TIM4) peuvent tolérer un léger jitter car elles gèrent des tâches moins critiques.

3 Décomposition préliminaire de l'application

3.1 Flot de données

Le diagramme de flot de données (DFD) suivant illustre le traitement des données depuis l'acquisition des capteurs jusqu'à l'affichage et la sauvegarde.

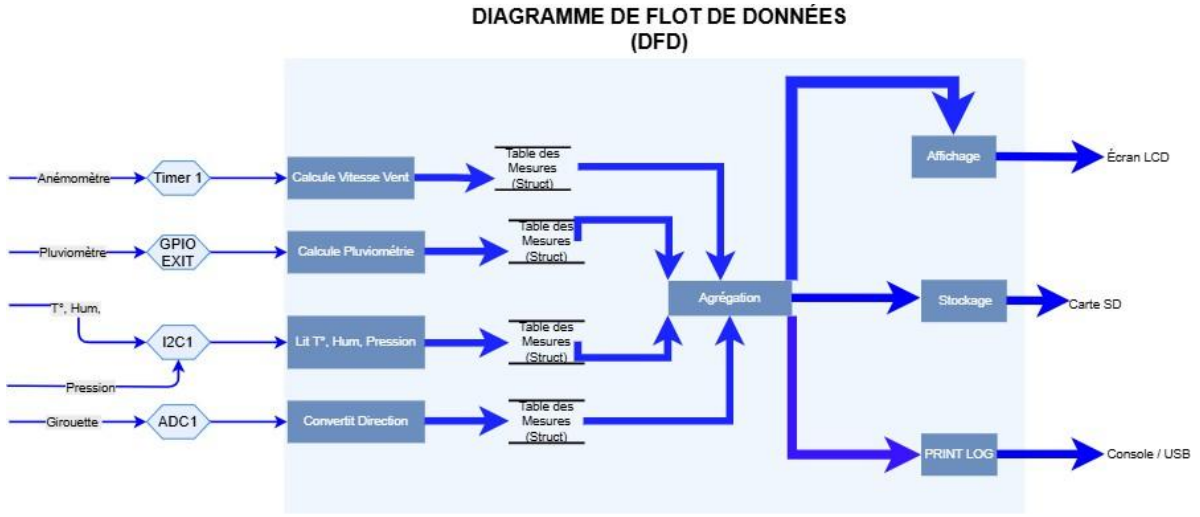


FIGURE 2 – Diagramme de flot de données (DFD) - Traitement des données du système

Le flot de données suit l'architecture suivante :

1. **Acquisition** : Lecture des capteurs via I2C (HTS221, LPS22HH), ADC (girouette), TIM Input Capture (anémomètre), EXTI (pluviomètre)
2. **Traitement** : Conversion des valeurs brutes en unités physiques (température en °C, pression en hPa, vitesse en km/h, etc.)
3. **Agrégation** : Stockage des données dans des buffers d'historique (tempHistory, humHistory, rainHistory, windHistory)
4. **Affichage** : Mise à jour de l'écran LCD selon la page courante
5. **Sauvegarde** : Écriture périodique sur la carte SD au format CSV

Le diagramme montre clairement la séparation entre les processus de calcul (Calcule Vitesse Vent, Calcule Pluviométrie, Lit T°, Hum, Pression, Convertit Direction), le stockage temporaire dans les structures de données, et les processus de sortie (Affichage, Stockage, PRINT LOG).

3.2 Flot d'événements

Le diagramme de flot d'événements (DFE) suivant illustre comment les événements d'interruption déclenchent les différentes actions du système via la machine d'états centrale.

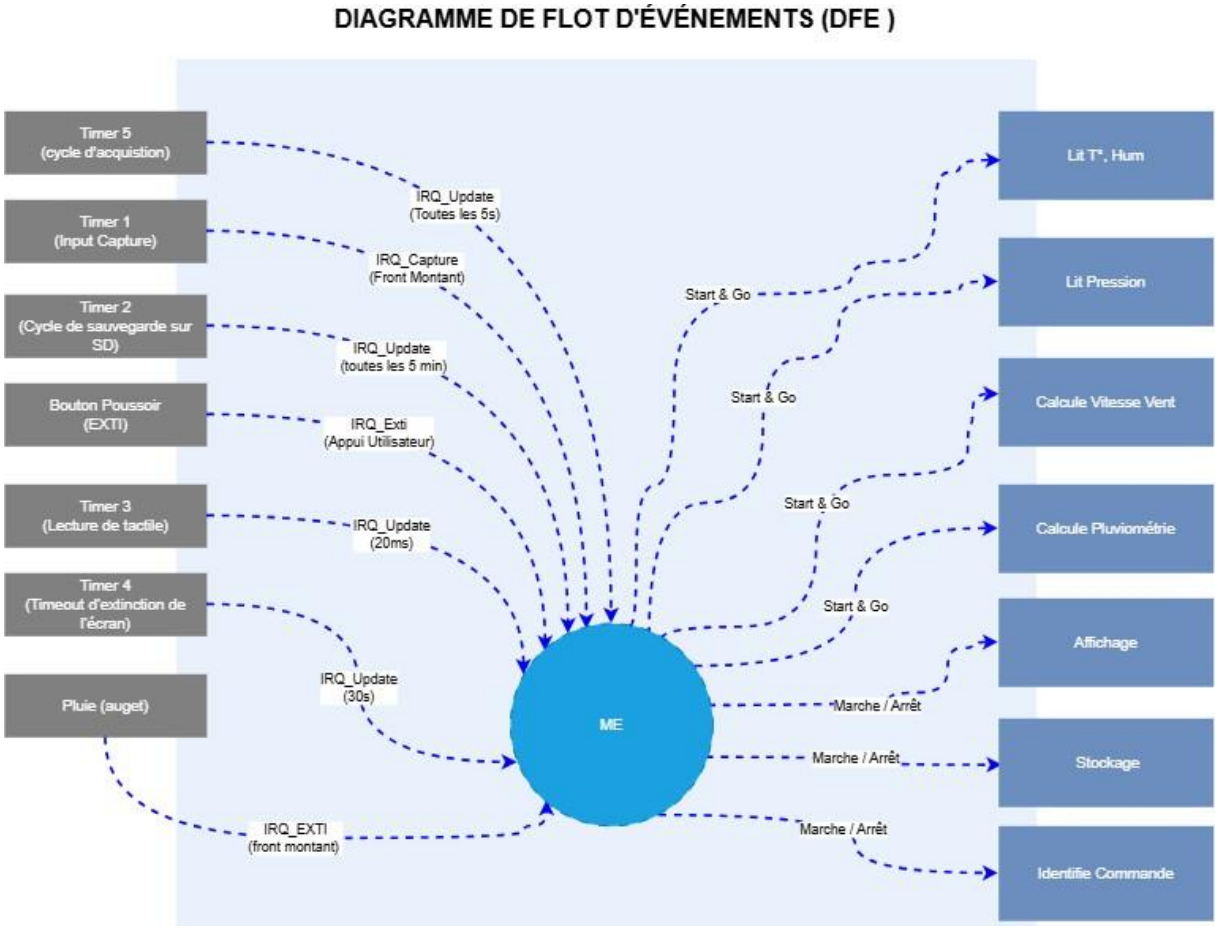


FIGURE 3 – Diagramme de flot d'événements (DFE) - Architecture événementielle du système

Le système est piloté par des événements générés par les interruptions :

- **TIM5** : Déclenche l'acquisition périodique des capteurs (IRQ_Update toutes les 5s)
- **TIM2** : Déclenche la sauvegarde périodique sur SD (IRQ_Update toutes les 5 min)
- **TIM4** : Déclenche la mise en veille de l'écran après 60 secondes (IRQ_Update après 30s)
- **TIM7** : Déclenche la mise à jour des LEDs d'état
- **TIM3** : Déclenche la lecture du touchscreen (IRQ_Update toutes les 20 ms)
- **EXTI0** : Réveil du système (IRQ_Exti sur appui bouton utilisateur)
- **EXTI15** : Détection de précipitations (IRQ_EXTI sur front montant du pluviomètre)
- **TIM1 IC** : Capture des impulsions de l'anémomètre (IRQ_Capture sur front montant)

Le diagramme montre que tous les événements convergent vers la machine d'états centrale (ME), qui orchestre ensuite les différentes actions (Lit T°, Hum, Lit Pression, Calcule Vitesse Vent, Calcule Pluviométrie, Affichage, Stockage, Identifie Commande).

3.3 Machine d'états

Le diagramme de machine d'états principale suivant illustre le comportement complet du système, depuis l'initialisation jusqu'aux différents modes opérationnels.

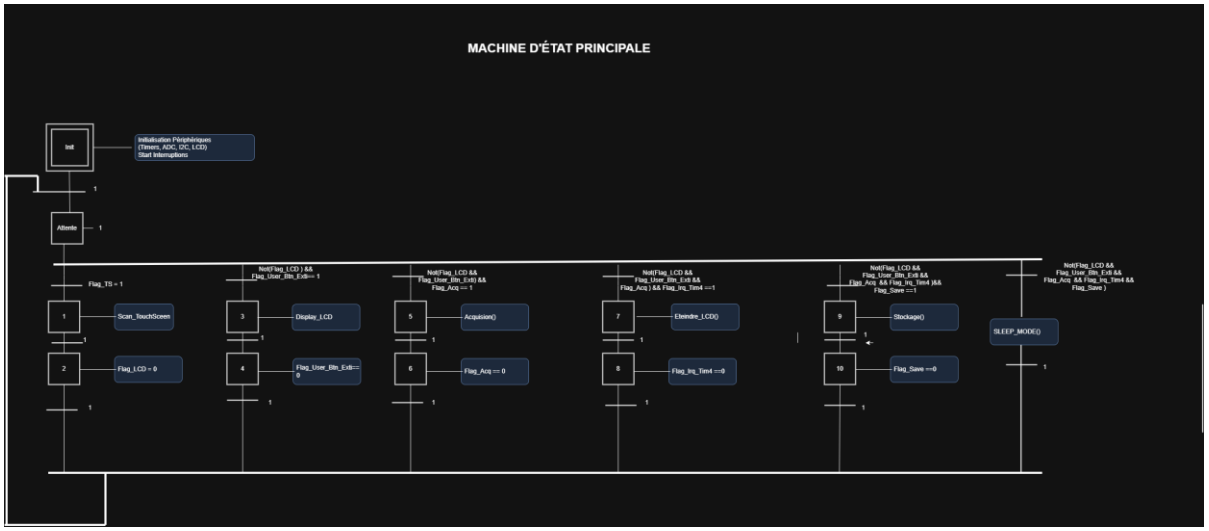


FIGURE 4 – Machine d'états principale - Comportement du système

Le système fonctionne selon une machine d'états gérée dans la boucle principale :

État INIT Initialisation des périphériques et affichage de l'écran d'accueil

État ATTENTE État d'attente où le système évalue les différents flags pour déterminer l'action à effectuer

Path 1 - Scan Touchscreen Lorsque Flag_TS = 1, le système scanne le touchscreen et réinitialise Flag_LCD

Path 2 - Display LCD Lorsque l'écran n'est pas éteint et qu'un appui bouton est détecté, le système met à jour l'affichage

Path 3 - Acquisition Lorsque Flag_Acq = 1, le système effectue l'acquisition des capteurs

Path 4 - Extinction LCD Lorsque Flag_Irq_Tim4 = 1, le système éteint l'écran après le timeout

Path 5 - Stockage Lorsque Flag_Save = 1, le système sauvegarde les données sur la carte SD

Path 6 - Sleep Mode Si aucun flag n'est actif, le système entre en mode SLEEP pour économiser l'énergie

Les transitions entre états sont déclenchées par :
— Sélection du temps d'acquisition (page 0 → page 1)
— Timer de veille écran (ACQUISITION → IDLE)

— Touch ou bouton utilisateur (IDLE → ACQUISITION)
— Timer de sauvegarde SD (ACQUISITION → SAVE_SD → ACQUISITION)

Le diagramme montre clairement l'architecture événementielle non-bloquante, où chaque action est conditionnée par des flags définis par les interruptions, et où le système retourne toujours à l'état ATTENTE pour évaluer les prochains événements.

4 Développement et validation unitaire

4.1 Capteur de direction du vent (Girouette)

4.1.1 Implémentation

Le capteur de direction du vent utilise l'ADC2 en mode polling pour lire la tension générée par le potentiomètre rotatif. Une table de correspondance convertit la tension en direction cardinale.

```
void Read_ADC2_Channel1() {
    HAL_ADC_Start(&hadc2);
    if (HAL_ADC_PollForConversion(&hadc2, 1000) == HAL_OK) {
        uint32_t adcValue = HAL_ADC_GetValue(&hadc2);
        voltage = (adcValue / 4095.0) * 3.3; // Conversion en
            volts
        HAL_ADC_Stop(&hadc2);
    }
    direction = getWindDirection(voltage);
}
```

Listing 1 – Fonction de lecture de la girouette

La fonction getWindDirection() utilise une table de correspondance basée sur les valeurs de résistance du potentiomètre :

```
char* getWindDirection(float voltage) {
    if ((voltage >= 0.7) & (voltage <= 0.9)) return "OUEST";
    else if ((voltage > 0.9) & (voltage <= 1.35)) return "NORD-
        OUEST";
    else if ((voltage > 1.35) & (voltage <= 1.55)) return "NORD";
    else if ((voltage > 2.2) & (voltage <= 2.4)) return "NORD-EST";
    else if ((voltage > 3.03) & (voltage <= 3.23)) return "EST";
    else if ((voltage > 2.86) & (voltage <= 3.02)) return "SUD-EST"
        ;
    else if ((voltage > 2.66) & (voltage <= 2.86)) return "SUD";
    else if ((voltage > 1.8) & (voltage <= 2.0)) return "SUD-OUEST"
        ;
    else return "Inconnu";
}
```

Listing 2 – Table de correspondance tension/direction

4.1.2 Validation

La validation a été effectuée en testant différentes orientations de la girouette et en vérifiant la cohérence des directions affichées avec les valeurs de tension mesurées.

4.2 Capteur de quantité de pluie (Pluviomètre)

4.2.1 Implémentation

Le pluviomètre génère une interruption EXTI à chaque basculement. Un compteur d'interruptions est incrémenté avec un mécanisme de débouncing logiciel.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin) {
    if (GPIO_Pin == GPIO_PIN_15) {
        float tick_actuel = HAL_GetTick();
        if ((tick_actuel - 0) > 1000) {    // Debouncing 1 seconde
            interrupt_count++; // Incrementer le compteur
        }
    }
    HAL_ResumeTick();
}
```

Listing 3 – Callback d'interruption pour le pluviomètre

La conversion en millimètres est effectuée selon la formule :

```
void detect_pluie() {
    rainfallAmount = interrupt_count * 0.2794;  // Calcul en mm
}
```

Listing 4 – Calcul de la quantité de pluie

4.2.2 Validation

Le capteur a été validé en simulant des basculements manuels et en vérifiant que le compteur s'incrémente correctement avec le débouncing approprié.

4.3 Capteur de vitesse du vent (Anémomètre)

4.3.1 Implémentation

L'anémomètre utilise le mode Input Capture du TIM1 pour compter les impulsions générées par la rotation de l'hélice.

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim) {
    if (htim->Instance == TIM1 &&
        htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1) {
        tick_count++;  // Incremente le compteur de ticks
    }
}
```

Listing 5 – Callback de capture pour l'anémomètre

La conversion en vitesse (km/h) est effectuée périodiquement :

```
void Get_Wind_Speed() {
    float ticks_per_second = tick_count;
    tick_count = 0;  // Reinitialisation pour la prochaine mesure
    windSpeedKmph = (ticks_per_second / 5) * 2.4; // Conversion en
        km/h
}
```

Listing 6 – Calcul de la vitesse du vent

4.3.2 Validation

La validation a été effectuée en faisant tourner l'hélice manuellement et en vérifiant que la vitesse calculée correspond aux rotations observées.

4.4 Capteur de température et d'humidité (HTS221)

4.4.1 Implémentation

Le capteur HTS221 communique via I2C1. Les données sont lues en vérifiant le registre de statut pour s'assurer que de nouvelles données sont disponibles.

```
void get_grandeur_values_sensor_hts221() {
    hts221_reg_t reg;
    hts221_status_get(&dev_ctx, &reg.status_reg);

    if (reg.status_reg.h_da) {
        // Lecture de l'humidit
        memset(&data_raw_humidity, 0x00, sizeof(int16_t));
        hts221_humidity_raw_get(&dev_ctx, &data_raw_humidity);
        humidity_perc = linear_interpolation(&lin_hum,
           data_raw_humidity);
        if (humidity_perc < 0) humidity_perc = 0;
        if (humidity_perc > 100) humidity_perc = 100;
        currentSensorData.hum = humidity_perc;
    }

    if (reg.status_reg.t_da) {
        // Lecture de la temp rature
        memset(&data_raw_temperature, 0x00, sizeof(int16_t));
        hts221_temperature_raw_get(&dev_ctx, &data_raw_temperature)
           ;
        temperature_degC = linear_interpolation(&lin_temp,
           data_raw_temperature);
        currentSensorData.temp = temperature_degC;
    }
}
```

Listing 7 – Lecture des données HTS221

4.4.2 Validation

La validation a été effectuée en comparant les valeurs mesurées avec un capteur de référence et en vérifiant la cohérence des variations lors de changements de température et d'humidité.

4.5 Capteur de pression (LPS22HH)

4.5.1 Implémentation

Le capteur LPS22HH communique également via I2C1. La lecture s'effectue en vérifiant le registre de statut.

```
void        get_values_pressure_sensor_lps22 hh ( void )       {
    lps22hh_read_reg(&dev_ctx, LPS22HH_STATUS, (uint8_t *)&reg, 1);

    if (reg.status.p_da) {
        memset(&data_raw_pressure, 0x00,  sizeof(uint32_t));
        lps22hh_pressure_raw_get(&dev_ctx, &data_raw_pressure);
        pressure_hPa = lps22hh_from_lsb_to_hpa(data_raw_pressure);
    }
}
```

Listing 8 – Lecture des données LPS22HH

4.5.2 Validation

La validation a été effectuée en comparant les valeurs mesurées avec les données météorologiques locales et en vérifiant la cohérence des variations.

5 Développement et validation des éléments d'assemblage

5.1 Agrégation des mesures

L'agrégation des mesures est effectuée dans la boucle principale lors de l'interruption TIM5. Les données sont stockées dans des buffers d'historique pour une sauvegarde ultérieure.

```
if(TimerFlag_Acquisition == 1) {
    // Mise    jour de l'affichage
    if(page == 1) {
        show_sensors();
    }
    else if(page == 2) {
        show_rain();
    }

    // Stockage dans les buffers d'historique
    if(sampleIndex < MAX_SAMPLE_COUNT) {
        tempHistory[sampleIndex] = currentSensorData.temp;
        humHistory[sampleIndex] = currentSensorData.hum;
        rainHistory[sampleIndex]   =   rainfallAmount;
        windHistory[sampleIndex]  =  windSpeedKmph;
        sampleIndex++;
    }
    else {
        sampleIndex = 0;   // R initialisation si buffer satur
```

```
20      }
21      TimerFlag_Acquisition = 0;
22  }
```

Listing 9 – Agrégation des données dans la boucle principale

5.2 Interface Homme-Machine

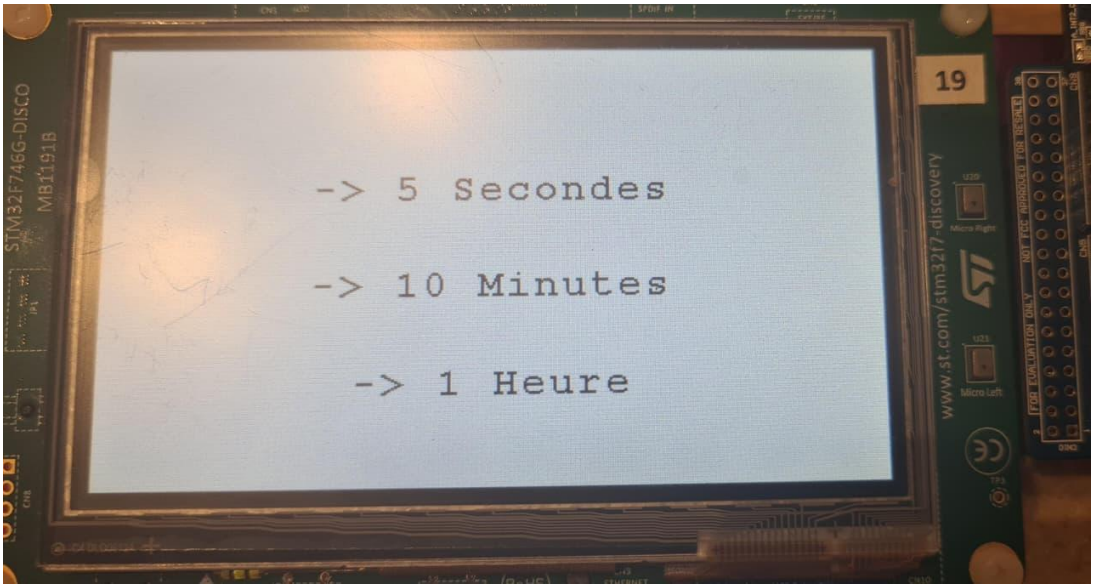
L'interface utilisateur a été conçue pour offrir une expérience fluide et intuitive, exploitant pleinement les capacités tactiles de l'écran LCD de la carte STM32F746NG Discovery (résolution de 480x272 pixels). Le design final adopte un style minimaliste et moderne, privilégiant la clarté de l'information et la rapidité de lecture. Le choix d'un fond gris clair et blanc assure une lisibilité maximale, même dans des conditions de luminosité variable.

5.2.1 Design et Ergonomie

L'ergonomie de l'application repose sur une navigation simplifiée entre trois écrans principaux, chacun dédié à une fonction spécifique du système. L'utilisation de polices sans-serif et d'éléments graphiques épurés renforce l'aspect professionnel de l'instrumentation.

Écran 1 : Configuration du système

Le premier écran permet à l'utilisateur de définir les paramètres opérationnels de la station. Il présente un menu vertical épuré pour la sélection de la période d'acquisition des données. Les options proposées sont calibrées pour répondre à différents besoins d'analyse : "5 Secondes" pour un suivi haute fréquence, "10 Minutes" pour une observation standard, ou "1 Heure" pour une surveillance à long terme.



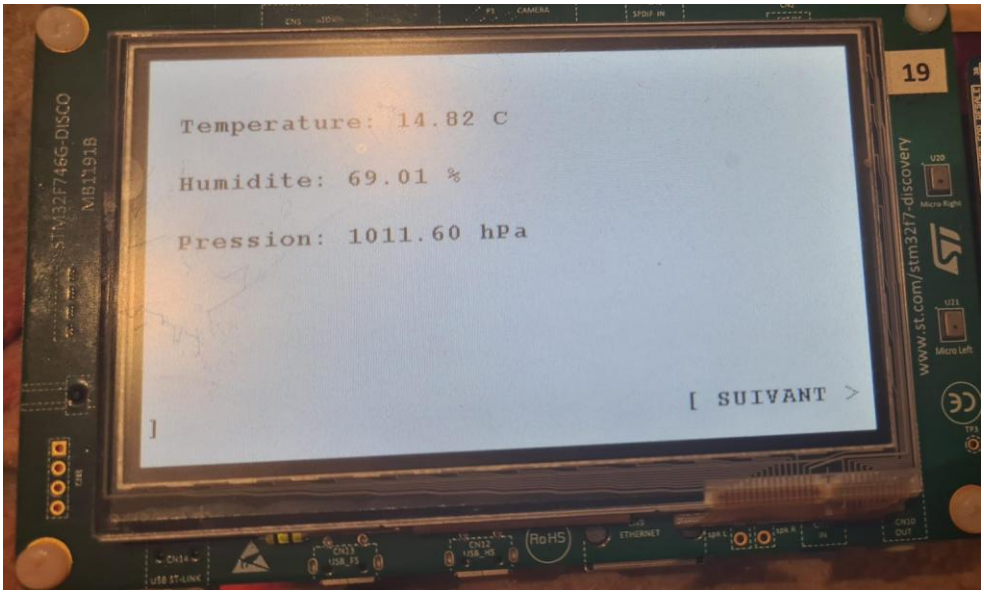
Écran 2 : Dashboard des Capteurs Internes

La deuxième page de l'interface, illustrée ci-dessus, constitue le tableau de bord principal des conditions atmosphériques intérieures. Elle affiche en temps réel les données collectées par les capteurs I2C situés sur la carte d'extension :

- Température : Mesurée par le capteur HTS221, affichée en degrés Celsius (°C) avec une précision de deux décimales.
- Humidité : Également issue du HTS221, affichée en pourcentage (%).
- Pression : Mesurée par le capteur LPS22HH, affichée en hectopascals (hPa).

Nous avons opté pour un design minimaliste avec une typographie sombre sur fond clair. Ce choix ergonomique maximise le contraste et la lisibilité, permettant une lecture rapide des valeurs flottantes même dans des conditions de luminosité variable.

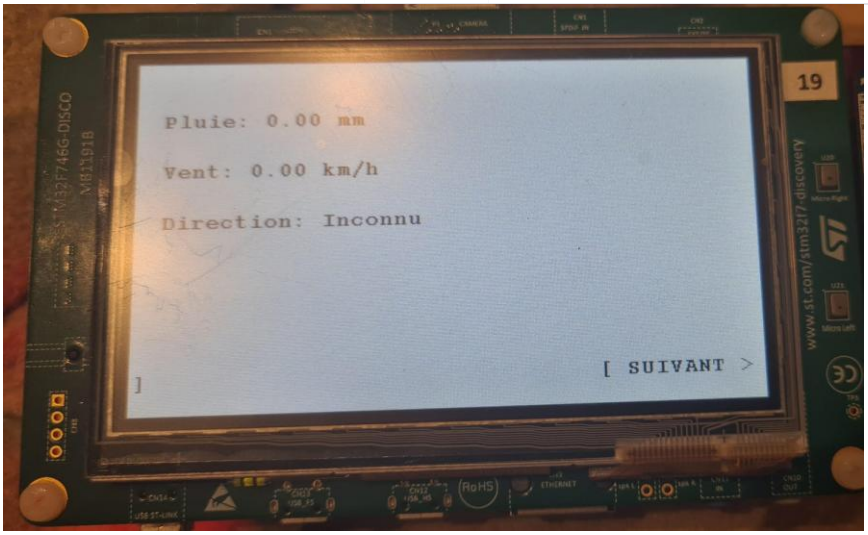
En bas à droite, le bouton tactile **[SUIVANT >]** invite l'utilisateur à naviguer vers la page suivante dédiée aux capteurs extérieurs (Pluie et Vent), assurant ainsi une fluidité dans l'expérience utilisateur."



Écran 3 : Données Extérieures

Le troisième écran est dédié aux capteurs mécaniques extérieurs. Les informations y sont présentées sous forme d'une liste textuelle claire et structurée, facilitant la lecture des données environnementales complexes.

Paramètre	Unité	Description
Pluie	mm	Cumul des précipitations mesuré par le pluviomètre
Vent	km/h	Vitesse instantanée calculée par l'anémomètre
Direction	Cardinal	Orientation du vent déterminée par la girouette



5.2.2 Gestion logicielle de l'affichage

L'IHM est pilotée par une tâche périodique (TIM3) cadencée à 20 ms pour assurer la réactivité du touchscreen. La bibliothèque graphique gère le rafraîchissement sélectif des zones de texte et des barres de progression pour minimiser la charge CPU et éviter les scintillements.

5.2 Sauvegarde sur carte SD

5.2.1 Implémentation FatFS

La sauvegarde utilise la bibliothèque FatFS pour gérer le système de fichiers sur la carte SD. Les données sont écrites au format CSV avec horodatage RTC.

```
void StoreSensorBatch(float *tempData, float *humData,
                      float *rainData, float *windData,
                      uint16_t sampleCount) {
    char buffer[100];
    UINT byteswritten;
    FRESULT res;
    uint16_t i;

    // Montage du syst me de fichiers
    if (f_mount(&SDFatFS, (TCHAR const *)SDPath, 0) != FR_OK) {
        Error_Handler();
        return;
    }

    // Formatage (si n cessaire)
    if (f_mkfs((TCHAR const *)SDPath, FM_ANY, 0,
               workBuffer, sizeof(workBuffer)) != FR_OK) {
        Error_Handler();
        FATFS_UnLinkDriver(SDPath);
        return;
    }

    // Ouverture du fichier CSV
    if (f_open(&SDFile, "data.csv", FA_OPEN_ALWAYS | FA_WRITE) !=
    FR_OK) {
        Error_Handler();
        FATFS_UnLinkDriver(SDPath);
```

```
27          return;
28      }
29
30      //   criture  de l'en-t te
31      snprintf(buffer, sizeof(buffer),
32               "Year/Month/Day;Hour:Minute:Second;Temp;Hum;Pluie;
                    Speed\n");
33      f_write(&SDFile, buffer, strlen(buffer), (void *)&byteswritten)
            ;
34
35      //   criture   des donn es avec horodatage RTC
36      for (i = 0; i < sampleCount; i++) {
37          RTC_TimeTypeDef sTime;
38          RTC_DateTypeDef  sDate;
39
40          HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN);
41          HAL_RTC_GetDate(&hrtc, &sDate, RTC_FORMAT_BIN);
42
43          snprintf(  buffer    ,              sizeof(  buffer),
44                   "%04d/%02d/%02d;%02d:%02d:%02d;%f;%f;%f;%f\n",
45                   2000 + sDate.Year, sDate.Month, sDate.Date,
46                   sTime.Hours, sTime.Minutes, sTime.Seconds,
47                   tempData[i], humData[i], rainData[i], windData[i])
                        ;
48
49          f_write(&SDFile, buffer, strlen(buffer), (void *)&
                byteswritten);
50      }
51
52      f_close(&SDFile);
53      FATFS_UnLinkDriver(SDPath);
54  }
```

Listing 12 – Fonction de sauvegarde sur SD

5.2.2 Format des données

Les données sont sauvegardées au format CSV avec le format suivant :

```
Year/Month/Day;Hour:Minute:Second;Temp;Hum;Pluie;Speed
2025/12/23;14:30:15;24.5;65.2;0.0;12.3
```

5.3 Gestion de l'énergie

Le système implémente une gestion de l'énergie efficace :

- **Mode SLEEP** : Entrée en mode SLEEP lorsque aucun événement n'est en cours
- **Veille écran** : Extinction automatique de l'écran après 60 secondes d'inactivité
- **Réveil instantané** : Réveil immédiat via touchscreen ou bouton utilisateur

```
1   else {
2       // Mode veille : entr e en mode SLEEP pour    conomiser   l'
            nergie
```

```
3       HAL_SuspendTick();
4       HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFE
            );
5       HAL_ResumeTick();
6   }
```

Listing 13 – Mode SLEEP dans la boucle principale

5.4 Répartition des tâches et Gestion de Projet

La réussite de ce projet repose sur une organisation rigoureuse et une répartition claire des responsabilités techniques. L'équipe a été structurée en trois pôles complémentaires, permettant un développement parallèle des modules avant leur intégration finale.

5.4.1 Tableau de répartition des tâches

Le tableau suivant synthétise la distribution des travaux entre les membres du groupe :

Tâche affectée	Tâche effectuée	Responsable(s)	Notes / livrables
Implémentation (lecture + agrégation) capteurs I2C (Température, Humidité, Pression)	Oui	**Boungab + Brahim**	Drivers + lecture capteurs + variables/struct de mesures
Implémentation capteurs de pluviométrie	Oui	**Boungab + Samuel Dassi**	Mesure quantité pluie + traitement associé
Implémentation capteurs de vent (anémomètre)	Oui	**Ibrahim**	Mesure vitesse vent + calcul/traitement
Implémentation interface utilisateur (pages UI)	Oui	**Ibrahim** (+ support design : **Boungab + Samuel Dassi**)	Écrans, navigation, rendu final
Interaction écran tactile (Partie 1)	Oui	**Ibrahim**	Détection touch + actions principales
Interaction écran tactile (Partie 2)	Oui	**Ibrahim**	Navigation avancée / boutons / changement pages
Implémentation RTC	Oui	**Équipe**	Date/heure + utilisation côté affichage/enregistrement

Tâche affectée	Tâche effectuée	Responsable(s)	Notes / livrables
Implémentation carte SD	Oui	**Équipe**	Log/CSV + write/read + gestion FATFS
Schématisation (Board model / DFD / DFE / Machine d'état / GRAFCET)	Oui	**Boungab+Samuel**	Tous les diagrammes + modélisation
Rédaction rapport	Oui	**Boungab + Ibrahim**	Rédaction + mise en forme + synthèse résultats

5.4.2 Méthodologie de développement

Le projet a suivi un cycle de développement modulaire. Chaque membre a d'abord validé ses composants de manière unitaire (comme détaillé en section 4) avant de procéder aux phases d'assemblage. Cette approche a permis de limiter les régressions et de faciliter le débogage lors de la fusion des branches de code. La gestion des priorités d'interruption a été le point de synchronisation majeur entre les pôles "Drivers" et "Noyau" pour garantir la stabilité du système global.

6 Conclusion

Ce projet a permis de développer une station météorologique complète et fonctionnelle sur la plateforme STM32F746NG Discovery. L'architecture événementielle basée sur les interruptions garantit un fonctionnement non-bloquant et efficace, tandis que l'interface utilisateur tactile offre une expérience utilisateur intuitive.

Les principaux défis rencontrés ont été :

— La gestion des priorités d'interruption pour éviter les conflits
— L'implémentation d'une interface tactile réactive
— La gestion de la sauvegarde sur carte SD avec FatFS
— L'optimisation de la consommation énergétique

Les résultats obtenus montrent que le système est capable de :

— Acquérir des données de manière fiable et précise
— Afficher les informations en temps réel
— Sauvegarder les données de manière périodique
— Gérer efficacement l'énergie du système

7 Annexes

7.1 Structure du projet

Le projet est organisé selon l'architecture suivante :

```
SE_Project/
```

```
Core/
    Inc/              # Headers des périphériques HAL
    Src/              # Sources principales
         main.c       # Boucle principale et logique système
         IRQ_Callback.c    # Gestion des interruptions
Drivers/
    Templates/
        Sensors/   # Drivers des capteurs
            humidity.c/h
            pression.c/h
            pluviometre.c/h
        Screens/ # Interface utilisateur
             display_reg.c/h
    Users/            # Drivers utilisateur (HTS221, LPS22HH)
FATFS/                # Middleware FatFS
```

7.2 Configuration des timers

TABLE 4 – Configuration des timers

Timer	Période	Usage	Priorité
TIM1	Input Capture	Anémomètre	7
TIM2	Variable	Sauvegarde SD	1
TIM3	20 ms	Touchscreen	-
TIM4	60 s	Veille écran	2
TIM5	Configurable	Acquisition	-
TIM7	500 ms	LEDs état	1

7.3 Format CSV

Le format CSV utilisé pour la sauvegarde des données :

En-tête: Year/Month/Day;Hour:Minute:Second;Temp;Hum;Pluie;Speed
Exemple: 2025/12/23;14:30:15;24.50;65.20;0.00;12.30

Les champs sont séparés par des points-virgules (;) pour faciliter l'import dans des tableurs.