Mai 2024

Projet Pico Smart Fan

EPHEC, Technologies de l'Informatique, Electronique 2

Groupe J:

Gaspard Derruine

Maxime Malpica

Hugo Noppe

Dorian Kwizera

A circuit board with wires and a fan

Description automatically generated

Objectifs du projet

* mettre en oeuvre les connaissances théoriques acquises durant les cours, le cours d'électronique principalement
* acquérir des compétences en matière de gestion de projet
* apprendre à travailler en groupe
* travailler dans le but d'atteindre un résultat concret

Description du projet

Le groupe a proposé de développer un prototype de ventilateur intelligent, construit autour d'un microcontrolleur.

Les caractéristiques du projet:

* lire un capteur de température et l'afficher sur un afficheur 7 segments à 2 chiffres.
* utiliser 2 SN74LS47N pour piloter les chiffres du 7 segments
* lorsque la température est au-dessus d'une température de référence donnée, activer le ventilateur jusqu'à ce qu'elle revienne à celle-ci
* piloter un ventilateur simple 5V (moteur CC)
* faire varier la vitesse du ventilateur en fonction de la température (avec PWM)
* mettre à jour la valeur de la température de référence avec une interface web
* implémenter un affichage LCD 1602 pour afficher des informations supplémentaires
* alerte sonore lorsque la température dépasse MAX
* LEDs pour indiquer l'état de la température

Planning

Voici les étapes principales du projet:

* Janvier 2024 : présentation et démarrage du projet
* 21 Février : présentation idée de projet
* 18 Mars : rapport d'avancement
* Mars-Avril-Mai : travail de groupe
* 17 Mai : présentation du produit final

Les tâches détaillées sont reprises en Annexe 1.

Simulation

Pour démarrer le projet, nous avons effectués quelques simulations de circuits sur Wokwi, mais nous sommes rapidement arrivés aux limites de ce qu'on peut faire d'utile pour un projet comme le nôtre, où la partir code est relativement importante.

Le PCB

Le PCB a été développé sur l'outil Eagle, qui a demandé pas mal d'apprentissage. Le passage du schema logique au circuit a amené pas mal de surprises lorsque qu'on a dû intégrer les contraintes physiques.

A circuit board with many wires

Description automatically generated

Ici aussi, les ressources offertes par Internet (tutos, YouTube, OpenAI etc) ont été fortement consultées.

A computer screen shot of a circuit board

Description automatically generated

Les composants

Les composants utilisés pour ce projet:

* Raspberry Pico W
* 2 74LS47 et 2 7segments + résistances de protection 220 ohms
* dht11
* LCD1602
* buzzer piezzo
* 3 leds
* potentiomètre 2K (ADC)
* ventilateur DC 5V
* PModHB5 (contrôleur moteur DC)

Le code

Le code micropython a été réalisé de façon modulaire. Chaque composant (LEDs, 7segment + 74LS47, LCD, buzzer, moteur DC + PWM, potentiomètre ADC, etc) a son propre module python, ce qui permet de tester chaque composant individuellement, et de l'intégrer de façon très propre et très efficace dans le module principal du projet.

Le code est structuré de cette façon:

* main.py (main + wifi + webserver)
  + params.py (paramètres wifi)
  + 74LS47\_7segments.py (affichage 7segments)
  + lcd.py (LDC1602)
  + dht11.py (capteur temperature)
  + leds.py (leds)
  + buzzer.py (alarme sonore)

La partie la plus délicate a été la gestion évènementielle et les appels à des procédures asynchrones (que nous ne maîtrisons pas encore tout à fait).

Github

Notre repository sur github : <https://github.com/MonsieurLoki/Pico>

Ce repository contient:

* les schémas pour le PCB
* le code micropython
* la documentation

Le résultat final

Malgré les difficultés techniques et organisationnelles, nous sommes plutôt satisfaits du résultat final.

Notre projet fonctionne réellement et répond aux spécifications initiales. Nous avons appris énormément et avons plein d'idées sur la façon dont ce projet pourrait être continué, et comment un nouveau projet pourrait bénéficier de toute l'expérience acquise durant les semaines précédentes.

L'interface web ressemble à ceci :

A circuit board with wires and a fan

Description automatically generated A screenshot of a phone

Description automatically generated

Les difficultés rencontrées

Le PCB

Le design et la réalisation du PCB a été une expérience complètement nouvelle pour tous les membres du groupe. La mise en pratique des techniques vues au cours a demandé pas mal d'efforts et a été une expérience relativement stressante, entre autres à cause du fait qu'il est difficile de simuler le bon fonctionnement du PCB avec les outils à notre disposition, et le fait que dès que le schéma du PCB part vers le fabricant, il est très compliqué de corriger une erreur de design.

Le matériel

Les composants électroniques n'étant pas en nombre suffisant pour que tout le monde ait un exemplaire de chaque composant (et en particulier du PCB ), il a fallu se répartir les tâches autrement (contributions au design du PCB, à la documentation, au code, à l'implémentation des composants individuellement sur breadboard, etc). L'un de nous s'est consacré à l'intégration des composants sur le PCB et l'intégration du code.

Micropython

Python est un langage nouveau pour le plupart des membres du groupe et le niveau de chacun est différent. Grâce à notre connaissance des langages appris dans les cours précédents (C, Javascript), et avec l'aide de l'IA, il a cependant été possible de réaliser la plupart des objectifs du projet, et surtout d'apprendre énormément.

GitHub

Nous avons commencé à utiliser GitHub l'année dernière pour des projets individuels. Travailler en groupe avec GitHub, et surtout pour un projet en micropython sur un microcontroller, et moins évident, d'autant plus si tout le monde n'a pas tout le matériel à disposition.

Le travail de groupe

Contribuer à un projet de développement de software en groupe, comme la projet Formule du premier quadri, est une chose. Collaborer efficacement sur un projet matériel est plus compliqué, dû aux évidentes contraintes physiques.

Extension possible du projet

Avec quelques jours de plus, les fonctionnalités suivantes seraient facilement implémentables sur notre prototype :

* consulter l'état du système (température, etc.) via l'appli web
* contrôler manuellement le ventilateur (MARCHE, ARRÊT) via l'appli web
* potentiomètre (ADC) pour ajuster la température de référence (celui-ci est déjà prévu dans le design initial et soudé sur le PCB)

Références bibliographiques et autres

Ce projet a principalement été réalisé sur base des matières vues au cours, des labos et autres travaux pratiques préparatoires. Il n'y a donc pas vraiment de bibliographie à documenter.

Beaucoup de tutos ont été consultés, soit en version statique (wikis), soit en version vidéo (YouTube).

Raspberry Pico

* Documentation Raspberry Pico : <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>
* Instllation micropython sur Pico: <https://www.youtube.com/watch?v=GiT3MzRzG48&ab_channel=CoreElectronics>

Pico micropython:

* Cette ressource en particulier nous a beaucoup aidé car elle nous a permis de comprendre et de régler un problème délicat, à savoir la gestion d'évènements asynchrones sur un microcontroller en micropython :

<https://www.youtube.com/watch?v=AK8UYh7pMGM&t=1s&ab_channel=CoreElectronics>

* Buzzer : <https://www.youtube.com/watch?v=S4zPLBTC7_w>
* Web Server : [How To Setup a HTTP Web Server With Raspberry Pi Pico W (using WiFi)](https://www.youtube.com/watch?v=AK8UYh7pMGM&ab_channel=CoreElectronics)
* tuto [H Bridge Motor Control Basics Explained](https://www.youtube.com/watch?v=YYMsS50x1UY&ab_channel=TheEngineeringMindset)
* tuto controlling motor with Hbridge (moteur : ig220019X00015r, controlleur :
* ow to sound a buzzer : <https://www.freva.com/how-to-connect-a-buzzer-to-a-raspberry-pi-pico/>
* Hbride DC motor controller : <https://digilent.com/reference/pmod/pmodhb5/start>
* tuto controlling a DC motor : <https://how2electronics-com.translate.goog/dc-motor-fan-speed-controller-with-raspberry-pi-pico/?_x_tr_sl=fr&_x_tr_tl=en&_x_tr_hl=fr&_x_tr_pto=wapp>
* [PWM on the Raspberry Pi Pico with MicroPython](https://www.youtube.com/watch?v=hqudYm-a_aU)
* i2c : <https://electrocredible.com/raspberry-pi-pico-i2c-tutorial-examples/>
* LCD1602 : <https://electrocredible.com/raspberry-pi-pico-lcd-16x2-i2c-pcf8574-micropython/>

Annexe : liste détaillée des tâches

Tasks

Done

* describe project and features
* test components individually
  + dht-11 (temperature sensor)
  + pmod5hb (HBridge driver for a DC motor)
    - fan working
    - speed can be adjusted
  + buzzer

[](https://github.com/MonsieurLoki/Pico/blob/main/pict/buzzer.png)

* create github repository
* create first version PCB
* tools -> checks ERC (Electronic Rules Check) and DRC (Design Rules Check)
* add a ground plane

Todo

* make sure all group members have access to the github repo
* check everything (project, features, components, PBC, etc)
* in particular, check PCB
  + schematic : check that each component individually is connected properly (VCC, GND, GPIO)
  + board : check that all connections are done properly (top and bottom layers, no intersections, no hanging wire, etc)
  + optimize signals ( )
  + annotate pinheads on the board
  + ...
  + check with [https://eurocircuits.com](https://eurocircuits.com/)
* test components individually
  + 7-segments led + 74LS47
  + LCD1602 display
  + onboard temperature sensor
  + ADC (with variable resistor - potentiometer)
* develop micropython functions to control each component individually, with the GPIOs as planned on the PCB
  + def getValuePot(): (return value between -5 and 5)
  + def initWifi()
  + def initWebServer()
  + def getTemperature()
  + def adjustFan(speed)
  + def displayLCDmsg(msg)
  + def display2digits(value)
  + ...

PCB

To be put on the PCB :

* Pico W
* 2x 7-segments DIGIT
  + 7 connections to SN74LS47
  + GND
* SN74LS47N nr 1 (unités)
  + 7 connections to 7-segment
  + GPIO 2 (D), 3 (C), 4 (B), 5 (A) : 4 bits -> 1 digit
  + BI/RBO, RBI, LT connected to VCC
* SN74LS47N nr 2 (dizaines)
  + 7 connections to 7-segment
  + GPIO 6 (D), 7 (C), 8 (B), 9 (A) : 4 bits -> 1 digit
  + BI/RBO, RBI, LT connected to VCC
* fan hbridge
  + GPIO 14 (ENABLE)
  + GND
  + VCC
  + DIRection : VCC (only one direction for the fan)
* buzzer
  + GPIO 15
  + GND
* variable resistor
  + VCC
  + GND
  + ADC 2 (GPIO 28)
* LCD 1602
  + VCC
  + GND
  + SDA (GPIO 16)
  + SCL (GPIO 17)
* DHT-11 sensor
  + GPIO 18 (signal)
  + VCC (5V)
  + GND
* led red
  + GPIO 19
* led red
  + GPIO 20
* led red
  + GPIO 21
* reserve PINs (just in case)
  + 1 : GPIO 22
  + 2 : RUN
  + 3 : GPIO 26
  + 4 : GPIO 27
  + 5 : GPIO 28