ESET 369 LAB 8 REPORT

Student name	Kyle Rex and Brink Van Eeden
Group number	43
Lab Section	501
Lab session date	April 18, 2025
Lab instructor	Xin Zhuang

INTRODUCTION

This lab focuses on configuring and programming two embedded systems using the Raspberry Pi platform and MicroPython. The primary objective is to develop a functional understanding of digital input and output, I2C communication, and graphical output through practical applications. System A involves emulating the Sense HAT hardware on a virtual Raspberry Pi using Thonny IDE and creating an interactive dot movement game that utilizes joystick input and LED matrix display functions. System B transitions to physical hardware implementation by programming a Raspberry Pi Pico to control two LEDs in an alternating blinking pattern using a simple MicroPython script. Both systems emphasize modular code development, real-time interaction, and hardware-software integration, reinforcing key embedded systems concepts.

SYSTEM A

The purpose of System A is to correctly configure a virtual Raspberry Pi on Oracle VM VirtualBox using Thonny IDE and SenseHat display/joystick to create a simple game using a python code sequence. This game involves creating a program that begins with displaying a cyan dot in the top left corner of the LED matrix and a red dot in the middle of the LED matrix. The user should then be able to move the cyan dot using the SenseHat joystick to the red dot while leaving a trail of cyan dots. When the cyan dot reaches the red dot the SenseHat screen will change to display a yellow hashtag symbol (#) indicating the conclusion of the program. Lastly, if the enter key is pressed on SenseHat (middle key of joystick) the program should restart. Understanding which header pins correlated with the LED matrix through I2C communication is extremely important in this lab. The relevant pins that correlate with SDA1 and SCL1 can be seen in Table 1.

Table 1: Pin Number I2C

	A Header Pin Number
SDA1	3
SCL1	5

Using the code sequences found in: *Learning Embedded Systems with MSP430 FRAM microcontrollers* by B. Hur, as references the code sequence used for this system was created. One part of the system was coded and tested at a time to ensure that there were no issues with the code or the hardware. The Python code created uses the sense_emu library to simulate the Sense HAT functionality on a Raspberry Pi. It begins by importing the required libraries and setting up color constants for red, cyan, yellow, and black using RGB tuples. The initial positions of the cyan and

red dots are defined as (0, 0) and (4, 4) respectively. The draw_initial() function clears the LED matrix and places the cyan and red pixels at their respective starting positions. This function is used both at the start of the program and whenever the user chooses to reset the matrix using the middle joystick button. The main loop continuously listens for joystick events using sense.stick.get_events(). When a joystick event is detected and the action is 'pressed', the code checks the direction of movement. Depending on whether the user pressed up, down, left, or right, the cyan dot's coordinates are adjusted, making sure the movement stays within the 8x8 boundaries of the matrix. If the middle button is pressed, the cyan dot is reset to its starting position at (0, 0), and the initial state is redrawn by calling draw_initial(). After each movement, the program checks if the new position of the cyan dot matches the red dot's position. If a collision is detected, the draw_yellow_hash() function is called. This function simply displays a yellow "#" character on the matrix using the sense.show_letter() method, which briefly replaces the LED display with a large symbol. If no collision occurs, the program updates the display to reflect the new position of the cyan dot and ensures the red dot is still visible at its fixed location. A short sleep (time.sleep(0.1)) is included at the end of the loop to prevent excessive CPU usage and allow smoother joystick input handling. Overall, the code makes effective use of Sense HAT's joystick input handling, pixel control functions, and display utilities to create a simple interactive dot movement and collision detection program. The Thonny IDE coding environment used to run this code sequence can be seen in Figure 1. Each stage of the simple game can be seen in Figure 2, Figure 3, and Figure 4.

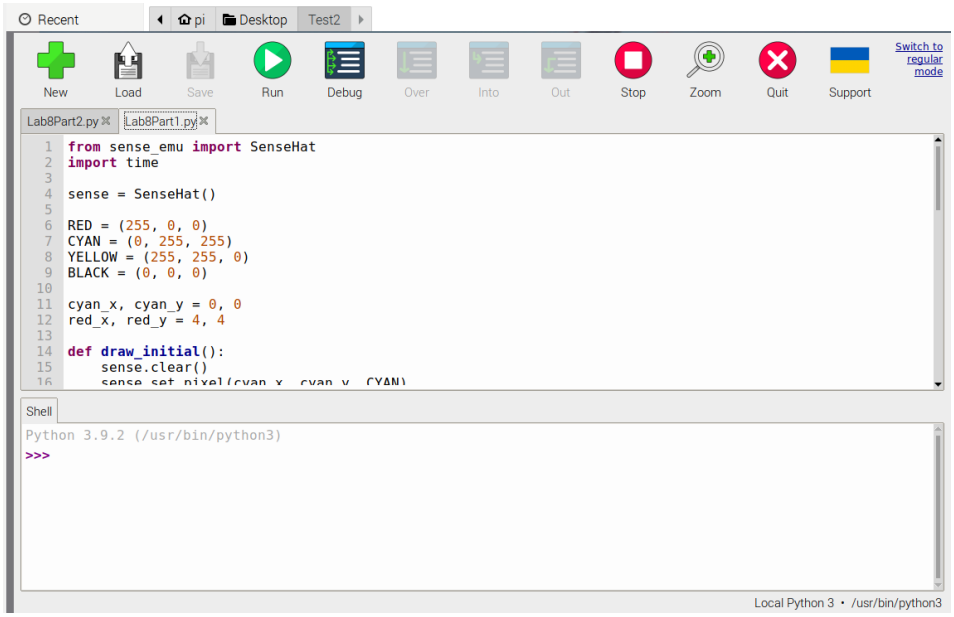

Figure 1: Thonny IDE Python Coding Environment

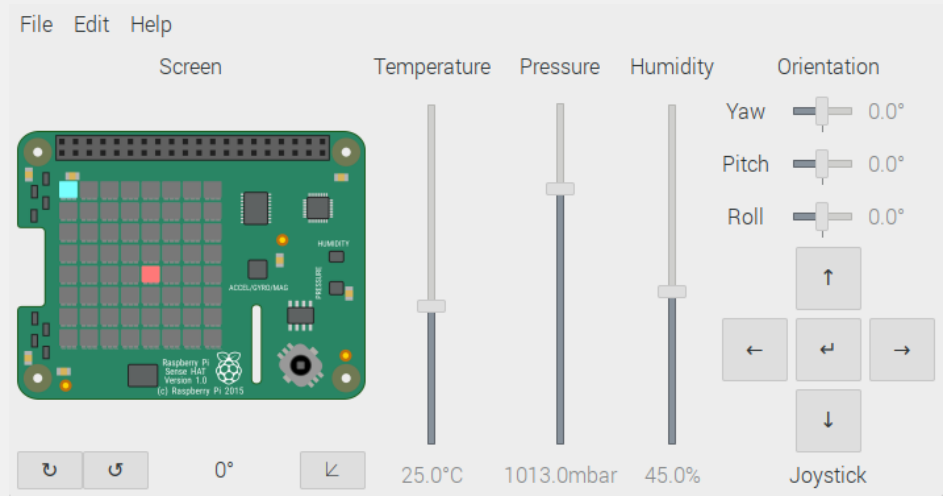

Figure 2: SenseHat Program Initial Display

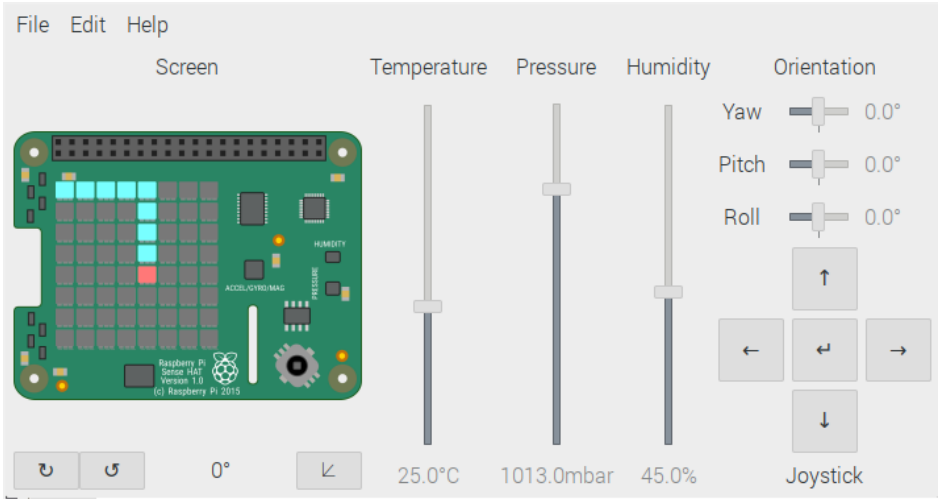

Figure 3: SenseHat Program Path Display

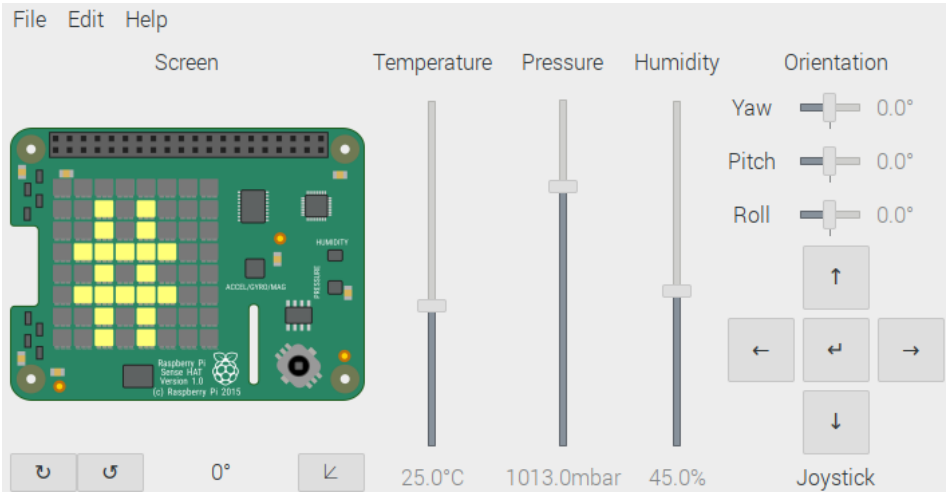

Figure 4: SenseHat Program Final Display

SYSTEM B

The purpose of System B is to correctly configure the Raspberry Pi (RPI) Pico to blink two LEDs alternately, at a reasonable speed, on the BH board LED bar using a micro-python code sequence. The first step in addressing this problem is making the physical board connections between the BH board and the Raspberry Pi (RPI) Pico. This can be done by connecting four specific pins together using male-to-female/male-to-male jumper wires and checking that all the DIP switches on the BH board are in the correct configuration. P38 of the RPI Pico must be connected to the top left pin, of the left set of pins, of the BH board where the microcontroller usually sits. P36 of the RPI Pico must be connected to the top right pin, of the right set of pins, of the BH board where the microcontroller usually sits. P19 of the RPI Pico must be connected to LED1 of the BH board LED bar. P20 of the RPI Pico must be connected to LED2 of the BH board LED bar.

Using the code sequences found in: *Learning Embedded Systems with MSP430 FRAM microcontrollers* by B. Hur, as references the code sequence used for this system was created. One part of the system was coded and tested at a time to ensure that there were no issues with the code or the hardware. The MicroPython script is designed to control two LEDs connected to GPIO pins 14 (P19) and 15 (P20). At the beginning of the program, the necessary modules Pin and sleep are imported from the machine and utime libraries, respectively. Two pins on the Pico are initialized as output pins using the Pin class: led1 is assigned to GPIO 14, and led2 to GPIO 15. The main functionality of the script lies within an infinite loop, which alternately turns the two LEDs on and off to create a blinking effect. In each cycle, led1 is first turned off (value(0)) while led2 is turned on (value(1)), and the program then pauses for half a second using sleep(0.5). After the pause, the LED states are swapped, led1 is turned on and led2 off, and the program again pauses for 0.5 seconds. This cycle continues indefinitely, causing the two LEDs to blink one after the other in an alternating pattern. The delay is slow enough for the alternating pattern to be clearly visible, as required in the lab. This simple blinking pattern demonstrates basic digital output control and timing in MicroPython on the Raspberry Pi Pico.

CONCLUSION

The completion of this lab successfully demonstrated essential skills in embedded systems programming and hardware configuration. System A highlighted the effective use of virtual environments and sensor emulation to simulate interactive applications involving joystick navigation and LED display logic. System B provided hands-on experience with GPIO control and timing operations using MicroPython on the Raspberry Pi Pico. The step-by-step testing approach for both systems ensured functionality and deepened comprehension of embedded software design and physical interfacing. Overall, the lab provided practical insights into the core principles of embedded systems and strengthened proficiency in debugging, modular programming, and system integration.

REFERENCES

B. Hur, *Learning Embedded Systems with MSP430 FRAM microcontrollers*, 2nd ed., 2023.

K. Rex, *ESET 369 Lab Report 7*, 2025.

K. Rex, *ESET 369 Lab Report 6*, 2025.

K. Rex, *ESET 369 Lab Report 5*, 2025.

K. Rex, *ESET 369 Lab Report 4*, 2025.

K. Rex, *ESET 369 Lab Report 3*, 2025.

K. Rex, *ESET 369 Lab Report 2*, 2025.

K. Rex, *ESET 369 Lab Report 1*, 2025.