Theme Report 1 - Observe

COMPENG 2DX3
Dr. Yaser Haddara

February 16th, 2025
Edison He (400449140)

Theme

The theme of this report is the ability for a system to observe the environment around it. In labs 1 to 3, the microprocessor is able to perceive and interact with the world around it. The theme of observation is applicable to smart homes and devices.The Google Nest thermostat, present in millions of homes including my own is a great example of observation. Its built-in sensors serve as inputs to the thermostat which allows it to monitor a home's temperature. Based on this data, it is able to output to the air conditioning or furnace to heat or cool the house to a certain temperature. The goal of the labs is an analysis of the system's input and output pins and their interactions with components in the circuit.

Background

A system's ability to observe is critical for smart devices. The goal of smart devices is to automate trivial tasks to save a user's time and money. However, this can only be done if the system has access to its surroundings, or to observe. Returning to the Google Nest example, automation of a home goes beyond temperature regulation. The device is able to monitor surroundings and analyze the user's behavior. It allows the user to save energy, and thereby money by reducing heating and cooling elements when the user is not present in the home. It provides the user comfort and seamless temperature automation by personalizing temperatures according to the day, time, and season and data it observes. These are not possible without real time monitoring or on-board analysis of data.

Labs 1 and 2 are straightforward but effective demonstrations of a system's ability to observe inputs and outputs. In lab 1, the microprocessor is used to read inputs by push buttons on a breadboard connected to pins of the microprocessor. Lab 2 is an extension of lab 1 with the use of Finite State Machines, where the microprocessor collects the inputs, processes them, and moves to the corresponding state in the Moore circuit. Lab 2 achieved a system which replicates a digital lock, but can be applicable in other situations such as vending machines, microwaves, computers, or anything with keypads. Lab 3 uses the same observation methodology to capture and convert analog signals to digital. The microprocessor stores an array of these digital signals which can be analyzed, or converted into a wave. This has a multitude of applications in sensors to capture sound, position, or light.

Theme Exemplars

For this report, Labs 1 and 2 will be used as theme examples. While lab 3 is a great example of observation, it was not completed in time due the closure of campus(Feb. 13, 2025) from the winter storm.

Lab 1

Lab 1 included 2 milestones which observed both input and output of the microprocessor. In milestone 1, outputs in the form of blinking LEDs were observed. This required the microcontroller to turn on and off LEDs, in the form of output 1 is on, and output 0 is off. A continuous loop containing a bitwise exclusive or representing toggle, and delay were introduced to continuously blink 2 leds, D0 and D1 seen below.

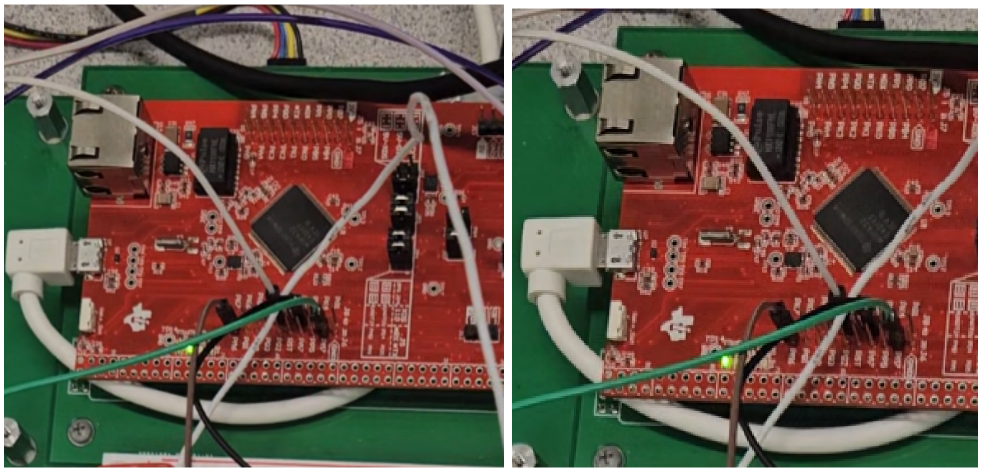


Figure 1: D1 and D0 Toggle

Milestone 2 built off of milestone 1, exploring a pushbutton input in relation to an LED. This was done with LED D1 and an active low button. When the microprocessor observes the button is not pressed, the active low circuit inputs 0, which the microprocessor reads, keeping the LED off. On the other hand, pressing the button results in an input of 1, observed by the microprocessor, which sends the input to D1, turning it on.

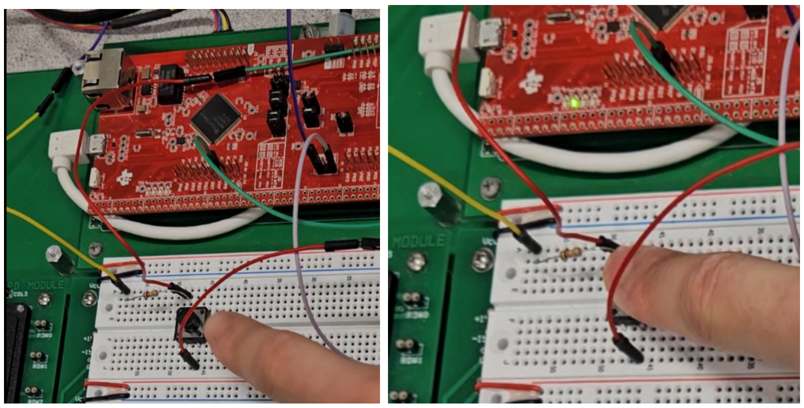
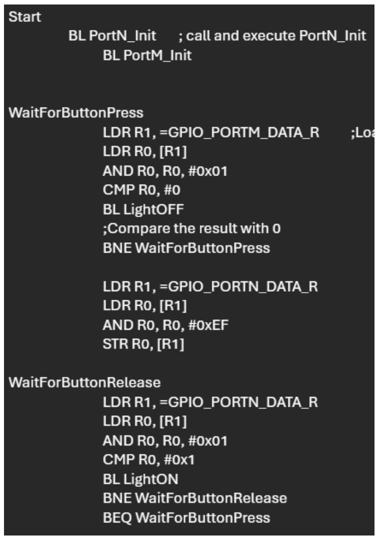


Figure 2: D1 ON and OFF with Button Press



```
Start
        BL PortN_Init     ; call and execute PortN_Init
            BL PortM_Init


WaitForButtonPress
            LDR R1, =GPIO_PORTM_DATA_R       ;Lo
            LDR R0, [R1]
            AND R0, R0, #0x01
            CMP R0, #0
            BL LightOFF
            ;Compare the result with 0
            BNE WaitForButtonPress

            LDR R1, =GPIO_PORTN_DATA_R
            LDR R0, [R1]
            AND R0, R0, #0xEF
            STR R0, [R1]

WaitForButtonRelease
            LDR R1, =GPIO_PORTN_DATA_R
            LDR R0, [R1]
            AND R0, R0, #0x01
            CMP R0, #0x1
            BL LightON
            BNE WaitForButtonRelease
            BEQ WaitForButtonPress
```

Figure 3: Portion of Code

Lab 2

In this lab, sequential circuit design was explored with the microprocessor. This sequential circuit design revolves around a lock with 5 states using Finite State Machine design. Inputs would be entered sequentially along with a clock pulse and the microprocessor would determine the correct state to move to based on the current state and input. When in the locked or intermediate states, D1 is on. In the unlocked state, D1 is turned off and D0 is turned on. A state diagram is provided below.

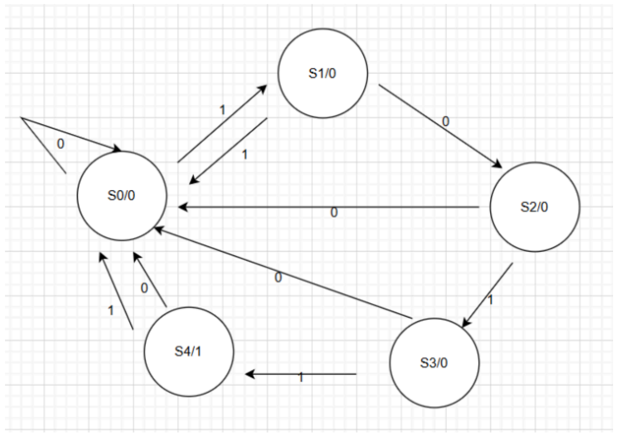


Figure 4: State Diagram of Synchronous Lock

This lab emphasized the decision making of the microprocessor based on the input observed. If the correct code is imputed, the microprocessor arrives in the unlocked state. If an incorrect input is observed during any state, the microprocessor returns to the initial locked state. A Moore circuit was used in this lab. This meant that if the microprocessor did not observe a manual clock signal change, the state and output were kept the same.

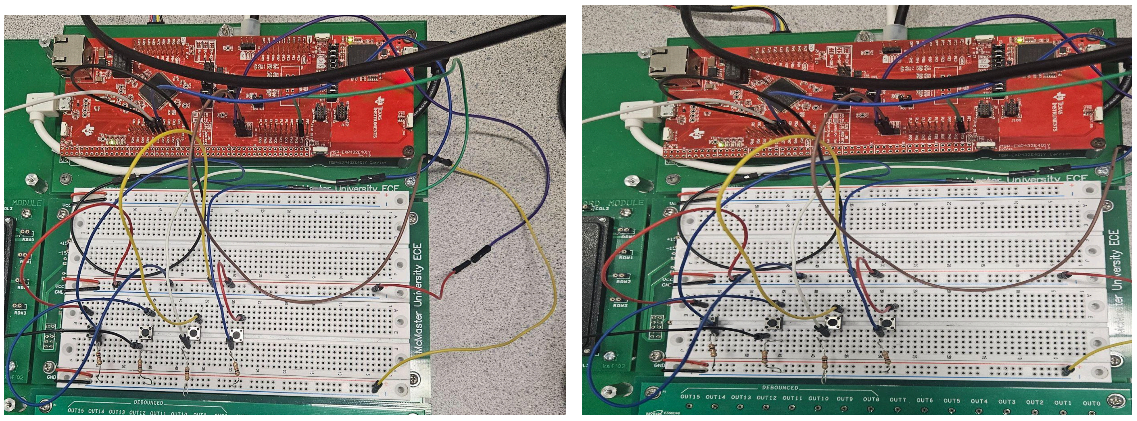
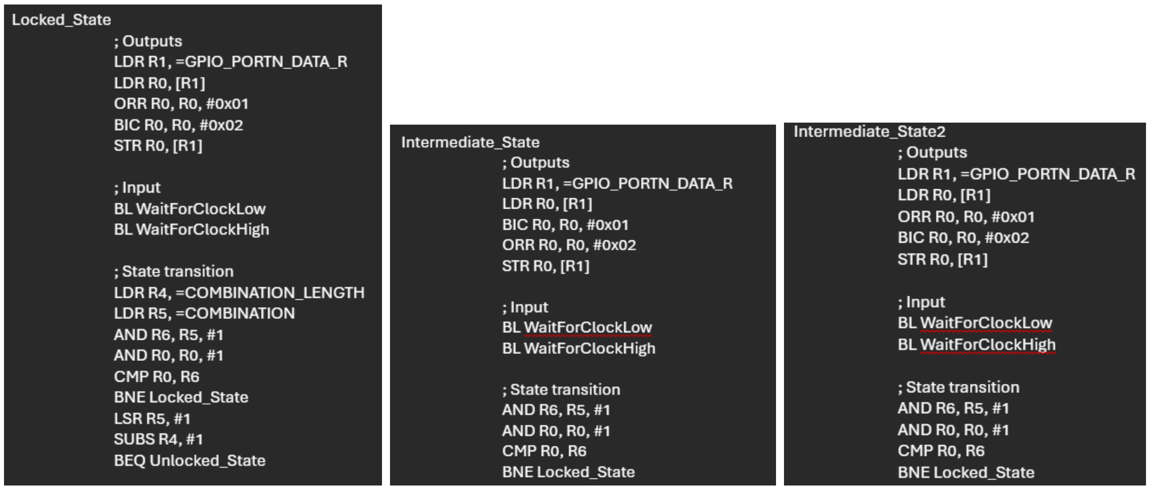


Figure 5: Locked and Unlocked States

Debugging

A debugging strategy was used in lab 2. As seen in Figure 4, there are 3 intermediate states besides the locked and unlocked states. As the output is only 1(D0) in the unlocked state, It is difficult to gauge the current state as each intermediate state along with the locked state has an output of 0(D1). Therefore, to verify the states are changing, we decided to toggle D0 and D1. This can be seen in the figure 6 below in which D0 and D1 are set or reset in addition to alternating depending on the state. For example, If the microprocessor is in the locked state, it has D0 off and D1 on. If a correct input is observed, it moves to the first Intermediate state. The user would know it is in the first Intermediate state as D1 turns off while D0 turns on. In addition, If an incorrect input is observed in the locked state, D1 stays on and D0 stays off. Once all states were accessed through this method, the code was revised to its original LED configuration in which every state turns D0 off and D1 on with exception to the reverse in the unlocked state.



```
Locked_State
                ; Outputs
                LDR R1, =GPIO_PORTN_DATA_R
                LDR R0, [R1]
                ORR R0, R0, #0x01
                BIC R0, R0, #0x02
                STR R0, [R1]

                ; Input
                BL WaitForClockLow
                BL WaitForClockHigh

                ; State transition
                LDR R4, =COMBINATION_LENGTH
                LDR R5, =COMBINATION
                AND R6, R5, #1
                AND R0, R0, #1
                CMP R0, R6
                BNE Locked_State
                LSR R5, #1
                SUBS R4, #1
                BEQ Unlocked_State
```

```
Intermediate_State
                ; Outputs
                LDR R1, =GPIO_PORTN_DATA_R
                LDR R0, [R1]
                BIC R0, R0, #0x01
                ORR R0, R0, #0x02
                STR R0, [R1]

                ; Input
                BL WaitForClockLow
                BL WaitForClockHigh

                ; State transition
                AND R6, R5, #1
                AND R0, R0, #1
                CMP R0, R6
                BNE Locked_State
```

```
Intermediate_State2
                ; Outputs
                LDR R1, =GPIO_PORTN_DATA_R
                LDR R0, [R1]
                ORR R0, R0, #0x01
                BIC R0, R0, #0x02
                STR R0, [R1]

                ; Input
                BL WaitForClockLow
                BL WaitForClockHigh

                ; State transition
                AND R6, R5, #1
                AND R0, R0, #1
                CMP R0, R6
                BNE Locked_State
```

Figure 6: Alternating LEDs to Verify a State Change

Synthesis

Labs 1 and 2 effectively illustrate the theme of observation in simple and complex tasks as the microprocessor alters its output based on the output observed. Lab 1 is the simplest version and can be seen as a proof of concept. Depending on the input observed, the microprocessor outputs the corresponding output to the LED. Lab 2 builds on this concept by introducing Finite State Machine Design. It makes decisions not only depending on the input observed, but on the current state. It decides either to advance to the next state on a correct input, or retreat to the base state on an incorrect input. This demonstrates the microprocessor to reason and act based on observations in the environment around it in addition to the ability to monitor its output and change if necessary. This concept overall is similar to the Google Nest Thermostat and how it alters its temperature based on the observed ambient temperature. If it detects the ambient temperature is too high compared to its target temperature, it lowers it and vice versa if the temperature is too low. If it detects no change when compared to its target temperature, it keeps the temperature the same.

Reflection

Lab 1 and 2 provided me with a solid understanding of observations in electronic systems and circuits. Although lab 1 is similar in nature, it provides a foundation for understanding observations in microprocessor systems. A takeaway from this lab is the active monitoring process, in which every clock cycle, the system must observe if there is a change in input. This process checks the input thousands of times per second, or more. It made me realize that simple tasks such as a change in input can be more complex than initially thought.

A takeaway from lab 2 is that not only does the microprocessor need to observe the environment, but the user must also observe the system. This is true in the debugging and testing process of milestone 2. As 4 of the 5 states have the same output, the microprocessor may know what state it is in, but the user does not. This is significant for user-friendly design as the system cannot aid the user if they do not understand how it works. In addition, the designer cannot fix issues if they cannot observe the faulty state. In the sequential lock design, It is impossible for the user/designer to know at which input sequence has failed. Additionally, if the user continues to enter the rest of the input as well as the correct input sequence after, the system will never reach the unlocked state until it is reset. This is not only a flaw for the user/designer, but the decision making of the system because it does not know when to properly reset. This is why we added the LED toggles to observe and verify that the states are changing and the system is working as intended. Overall, both labs were crucial to understanding the significance of observation to the decision making process of a system.