|  |
| --- |
| ECSE 421 – Embedded Systems  Laboratory #3 |

|  |
| --- |
| **Interrupts & I/O** |

|  |
| --- |
| Zichen Gao (261055991)  Presented to Prof. Jeremy Cooperstock  Due March 14th 2025 |

Contents

[Exercise 1: A Simple Doorbell 3](#_Toc191675119)

[**Task:** Write a program that rings the buzzer every time the button is pressed, using an interrupt and ISR to control the buzzer. You should not have anything running in the void loop() method, other than a delay as a placeholder. Implement a two-tone buzz (or feel free to get more creative with your chime!). 3](#_Toc191675120)

[**Question:** Will the delay() function work inside the ISR (interrupt service routine)? What about micros()? It may be helpful to try this out. Give an explanation for your answers. 3](#_Toc191675121)

[Exercise 2: The Smart Home 4](#_Toc191675122)

[**Question:** Is this system best modelled as a Mealy machine or Moore machine? Explain your choice. 4](#_Toc191675123)

[**Task:** Create a state diagram for the above specifications. Using your diagram, create the state-transition logic table. Include the current states, as well as the inputs and outputs. 4](#_Toc191675124)

[**Task:** Design the smart home controller in Arduino using switch cases, which should be representative of each state in your state diagram. We recommend creating functions to organize your code. 10](#_Toc191675125)

[Exercise 3: Door Security System 11](#_Toc191675126)

[**Task:** To begin, draw the state models for each component. 11](#_Toc191675127)

[**Task:** Using your Arduino and available sensors, design the controller as given in the specifications. 16](#_Toc191675128)

# Exercise 1: A Simple Doorbell

**Task:** Write a program that rings the buzzer every time the button is pressed, using an interrupt and ISR to control the buzzer. You should not have anything running in the void loop() method, other than a delay as a placeholder. Implement a two-tone buzz (or feel free to get more creative with your chime!).

*To generate blocking delays inside the ISR, I made use of the fact that the Serial monitor requires time to transmit which is dependent on the baud rate. Hence, we can estimate a delay by sending a certain number of characters over to the serial monitor. Assuming that the delay in ms produced by sending one character is about 50/BAUD\_RATE \* 1000, we can calculate that in order to generate a delay of X ms, we need to send out X\*BAUD\_RATE/(50000) characters.*

*See code and video for the rest of the exercise.*

**Question:** Will the delay() function work inside the ISR (interrupt service routine)? What about micros()? It may be helpful to try this out. Give an explanation for your answers.

*The delay() function in Arduino works by having a low priority interrupt count a timer. However, since our ISR will (likely be) of higher priority than the timer ISR, it will never count during an ISR and hence delay() does not work inside ISRs. Although the micros() function reads directly from a timer register, it unreliable behavior inside of ISRs.*

*According to this Stack Overflow post (*[*https://arduino.stackexchange.com/questions/22212/using-millis-and-micros-inside-an-interrupt-routine*](https://arduino.stackexchange.com/questions/22212/using-millis-and-micros-inside-an-interrupt-routine)*), the implementation of micros() looks something like the code below:*

unsigned long micros() {

unsigned long m;

uint8\_t oldSREG = SREG, t;

cli();

m = timer0\_overflow\_count;

t = TCNT0;

if ((TIFR0 & \_BV(TOV0)) && (t < 255))

m++;

SREG = oldSREG;

return ((m << 8) + t) \* (64 / clockCyclesPerMicrosecond());

}

*Note that the* timer0\_overflow\_count *variable, which is included in return value of the function, is updated by an ISR. This means that the micros() function will work fine as long as the delay is not too long, as we will miss the ISR that updates the* timer0\_overflow\_count *variable, which makes the micros() function reset to 0.*

# Exercise 2: The Smart Home

**Question:** Is this system best modelled as a Mealy machine or Moore machine? Explain your choice.

*Although one of the outputs of this system is to display the state of the system to the serial terminal, which would be more suitable for the Moore machine, most of the outputs are triggered by the inputs. For example, the heating is turned on or off depending on the value of the temperature sensor (input). Overall, this system is more suited for a Mealy machine, where outputs can be triggered directly by inputs rather than the state of the system.*

**Task:** Create a state diagram for the above specifications. Using your diagram, create the state-transition logic table. Include the current states, as well as the inputs and outputs.

*To begin, below is a brief description of all the inputs, outputs and variables associated with the state diagram (Figure 1). I chose to create a diagram with only two states to facilitate implementation and to reflect that fact that the main loop of the Arduino is always polling the sensors and ready to give many outputs (e.g., rolling down the shade and turning off the light) in a single loop. We want the system to go right back to monitoring as soon as it activates an output. Therefore, there are only two states: the ‘monitoring state’ where the system monitors all the sensors (temp, light, motion) and producing outputs based on the results read from the sensors, and the ‘menu state’ where the system is changing the parameters based on user input. Hence all the outputs are given in the state transitions (as a Mealy machine would do). In addition, the state-transition logic table (Table 1) is provided right below the state diagram.*

*Description of inputs:*

*Pure signals (absent or present):*

* *motion: Whether motion is detected using the motion detector.*
* *joystickHeld: Whether the joystick has been held for some time.*

*Other signals:*

* *light: {absent, some, alot} – Light level detected by the light sensor.*
* *T: {float} – Temperature reported by the temperature sensor.*
* *userX: {float} – User input for heating threshold temperature.*
* *userY: {float} – User input for cooling threshold temperature.*
* *userZ: {int} – User input for light timeout duration (in seconds).*

*Description of outputs:*

*Pure signals (absent or present):*

* *heatOn: Whether the heating system is turned on.*
* *heatOff: Whether the heating system is turned off.*
* *coolingOn: Whether the cooling system is turned on.*
* *coolingOff: Whether the cooling system is turned off.*
* *lightOn: Whether the lights are turned on.*
* *lightOff: Whether the lights are turned off.*
* *rollDownShade: Whether the shades are rolling down.*
* *rollUpShade: Whether the shades are rolling up.*
* *menuOpen: Whether the menu is open.*
* *menuClosed: Whether the menu is closed.*

*Other outputs:*

* *systemHeatState: {string} – Current state of the heating/cooling system (e.g., "heating", "cooling", "maintain temp").*
* *systemLightState: {string} – Current state of the lights (e.g., "light on", "light off").*
* *systemShadeState: {string} – Current state of the shades (e.g., "rolling down", "rolling up", "down", "up").*
* *systemMenuState: {string} – Current state of the menu (e.g., "menu open", "menu closed").*

*Description of variables:*

* *X: {float} – Heating threshold temperature (set by user or default).*
* *Y: {float} – Cooling threshold temperature (set by user or default).*
* *Z: {int} – Light timeout duration (set by user or default).*
* *s(t): {continuous, positive float} – Timer for light timeout.*
* *r(t): {continuous, positive float} – Timer for shade rolling down.*
* *u(t): {continuous, positive float} – Timer for shade rolling up.*
* *isShadeDown: {boolean} – Whether the shades are currently down (true) or up (false).*

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 1: *State diagram of the smart home system*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Current state** | **Condition** | **Next State** | **Outputs** | **Variables** |
| **Initial State** |  | Monitoring State |  | s(t)=0  s’(t)=0  r(t)=0  r’(t)=0  u(t)=0  u’(t)=0  X=20  Y=25  Z=10  isShadeDown=false |
| Monitoring State | T < X | Monitoring State | heatOn, coolingOff, systemHeatState=  “heating” |  |
| Monitoring State | T > X ^ T < Y | Monitoring State | heatOff,  coolingOff,  systemHeatState=  “maintain” |  |
| Monitoring State | T > Y | Monitoring State | heatOff, coolingOn, systemHeatState=  “cooling” |  |
| Monitoring State | s(t)=0 ^ light ^ motion | Monitoring State | lightOn,  systemLightState=  “light on” | s’(t)=1 |
| Monitoring State | s(t) > 0 ^ s(t) < 10 ^ motion | Monitoring  State |  | s(t)=0 |
| Monitoring State | s(t) > Z | Monitoring State | lightOff,  systemLightState=  “light off” | s’(t)=0  s(t)=0 |
| Monitoring State | isShadeDown=false ^ light=a lot  ^ r(t)=0 | Monitoring State | rollDownShade,  systemShadeState=  “rolling down” | r’(t)=1 |
| Monitoring State | r(t) > 8 | Monitoring State |  | r’(t)=0  r(t)=0  isShadeDown=true |
| Monitoring State | isShadeDown=true  ^ (light=a lot)  ^ u(t)=0 | Monitoring State | rollUpShade,  systemShadeState=  “rolling up” | u’(t)=1 |
| Monitoring State | u(t) > 10 | Monitoring State |  | u’(t)=0  u(t)=0  isShadeDown=false |
| Monitoring State | joyStickHeld | Menu Open | systemMenuState=  “menu open” |  |
| Menu Open | userX | Menu Open |  | X=userX |
| Menu Open | userY | Menu Open |  | Y=userY |
| Menu Open | userZ | Menu Open |  | Z=userX |
| Menu Open | (Y > X) | Menu Open | userInputError | X=20  Y=25 |
| Menu Open | joyStickHeld | Monitoring  State | systemMenuState=  “menu closed” |  |

Table 1: *State transition logic table*

**Task:** Design the smart home controller in Arduino using switch cases, which should be representative of each state in your state diagram. We recommend creating functions to organize your code.

*For this task, everything can be seen in the video or/and the source code.*

# Exercise 3: Door Security System

**Task:** To begin, draw the state models for each component.

*The state models for each component are shown in Figures 2-7 below.*

A diagram of a device

AI-generated content may be incorrect.

A diagram of a flowchart

AI-generated content may be incorrect.Figure 2: *Tamper Detector state model*Figure 3: *Presence detector state model*A screenshot of a computer

AI-generated content may be incorrect.Figure 4: *Keypad state model*A screenshot of a computer

AI-generated content may be incorrect.

Figure 5: *RGB LED state model*

A diagram of a system

AI-generated content may be incorrect.

Figure 7: *Buzzer state model*

A diagram of a computer

AI-generated content may be incorrect.

Figure 7: *Controller state model*

**Task:** Using your Arduino and available sensors, design the controller as given in the specifications.

*For this task, everything can be seen in the video or/and the source code.*