Building systems Using Microcontroller-Controlled I/O

Dr. Anu Aggarwal, University of Illinois at Urbana - Champaign

Anu Aggarwal is an Assistant Professor at the University of Illinois, Urbana Champaign. She secured her PhD in Electrical and Computer Engineering from the University of Maryland College Park under the supervision of Prof Robert W Newcomb in the area of N

Building systems using Microcontroller controlled I/O

This paper describes a novel microcontroller (MCU) based course that was added to the Electrical and Computer Engineering curriculum at our university.

1 Significance

Need for this course was felt in the department because several research faculty members wanted to integrate VLSI chips using MCU-based systems. To integrate custom VLSI chips into a system using MCU based control, one method is to build custom printed circuit board (PCB) to integrate components like opamp and buffers to read outputs from a chip. To send inputs to the chip, different pulse generator chips and voltage regulators could be integrated on the same PCB. To build a system out of several chips, MCU can be used to calculate inputs to the next chip based on outputs from the previous chip and algorithm designed in C on the MCU. Therefore, to achieve the requirements of our department, the course included C programming of the MCU, and custom PCB design. Hardware knowledge of MCU was also provided as part of the curriculum. Students also learned to integrate input-output circuits on the PCB. Even though main requirement at our university was for integrating VLSI chips, projects using commercially available chips were also assigned.

MCU based system design courses are offered at several universities. Our course was one of the few that integrated PCB design with MCU-based system design to build systems using custom and commercially available VLSI chips. This course can be especially useful for hobbyists, future entrepreneurs, and researchers.

2 Background

MCU based systems connect to the outside world via sensors and actuators. The MCU reads inputs from sensors, processes them, and sends outputs to actuators based on the algorithms coded by the designer. These interface devices can be integrated on a PCB for system design. This course provided experience in using an MCU (MSP 432) to interface with sensors and actuators and to integrate interface devices on a PCB for system design. The interface devices were both analog and digital that are commonly used for practical applications.

3 The course

a) Learning objectives/outcomes

Course objective was to use MSP 432 MCU and PCB in designing systems. Learning objectives included leveraging hardware knowledge about circuit design and the MCU and software knowledge about C programming language to build MCU based systems and to learn PCB design. The learning objectives were instilled through weekly lectures and labs. Knowledge acquired was tested using prelab and lab exercises and two end of term projects.

b) Curriculum

This course introduced students to fundamentals of MSP 432 MCU architecture (registers, memory types, signal bus, input/output ports) so that they could develop programs in C to interface with input devices like switches, keypad, temperature and touch sensors, display with LEDs and LCDs, digital to analog and analog to digital conversion, control motors

and actuators, MCU memory types and usage, MCU timer peripherals and MCU interrupt-driven timing. They were also introduced to PCB design and soldering so that they could build custom input-output boards for the system. The focus was on hands on experience in the lab to extend the theoretical knowledge gained from other courses, integrate hardware-software to develop projects with real life use. Class schedule is shown in table below.

Table 1 Class schedule for the course

Week	Topic	Lab	Goals
1	Introduction to MSP microcontroller, MSP architecture	LAB1: Integrated development and debug for programs on the MCU	Familiarize with the development environment for coding, debugging and programming the MCU.
2	Logic Functions in C and Bitwise manipulations in C (as required to set pins of the MCU) Manipulating GPIO	LAB 2 & 3: Interfacing switches and LEDs with MSP432	Learn how inputs (on the dev board) can be read and external outputs be controlled using MCU.
3	Interfacing with keypad	LAB4: Interfacing MCU using a keypad	Reading datasheets - the LED and push button circuits. Use Systick Timer to generate time intervals.
4	LCD displays	LAB5: LCD interface	Learn to use LCD screen with the MCU to print, scroll and float text floats.
5	Timers, PWM, DC motors servo solenoid, stepper motor control	LAB6: PWM to control LED intensity and DC motor speed	Control motor speed using the MCU: design electromechanical interface.
6	ADC	LAB7: Interfacing temperature and pressure sensor and LCD, building a weather station	Read analog sensors, use ADC module on chip, design algorithms to predict weather
7	IR communication	LAB8: Control traffic lights using IR sensor	Learn about IR communication and its application in remote controlled devices
8	Serial communication (UART)	LAB9: Configure UART using MCU and communicate with PC	Learn about interfacing of computer with MCU
9	Interrupts	LAB10: Use interrupts for Timer and IO	Learn and use interrupts to read data from external inputs and use for timer
10	PCB circuits, layout, testing and debug	LAB11: PCB layout	Learn PCB design, solder and debugging
11&12	Introduction to Projects	Work on Project 1	Learn to integrate several sensors into a system
13	Fall break	Fall break	-
14 &15	Work on Project	Work on Project 2	Learn to take inputs, design logic to process them and change interface based on the logic and inputs

Labs were conducted using a lab kit that is shown below. Names of the components in the kit are given in the list to the left. All resources required to complete projects or labs were contained in the kit. So, students could work on them from their home or in lab using the lab kit and a computer.

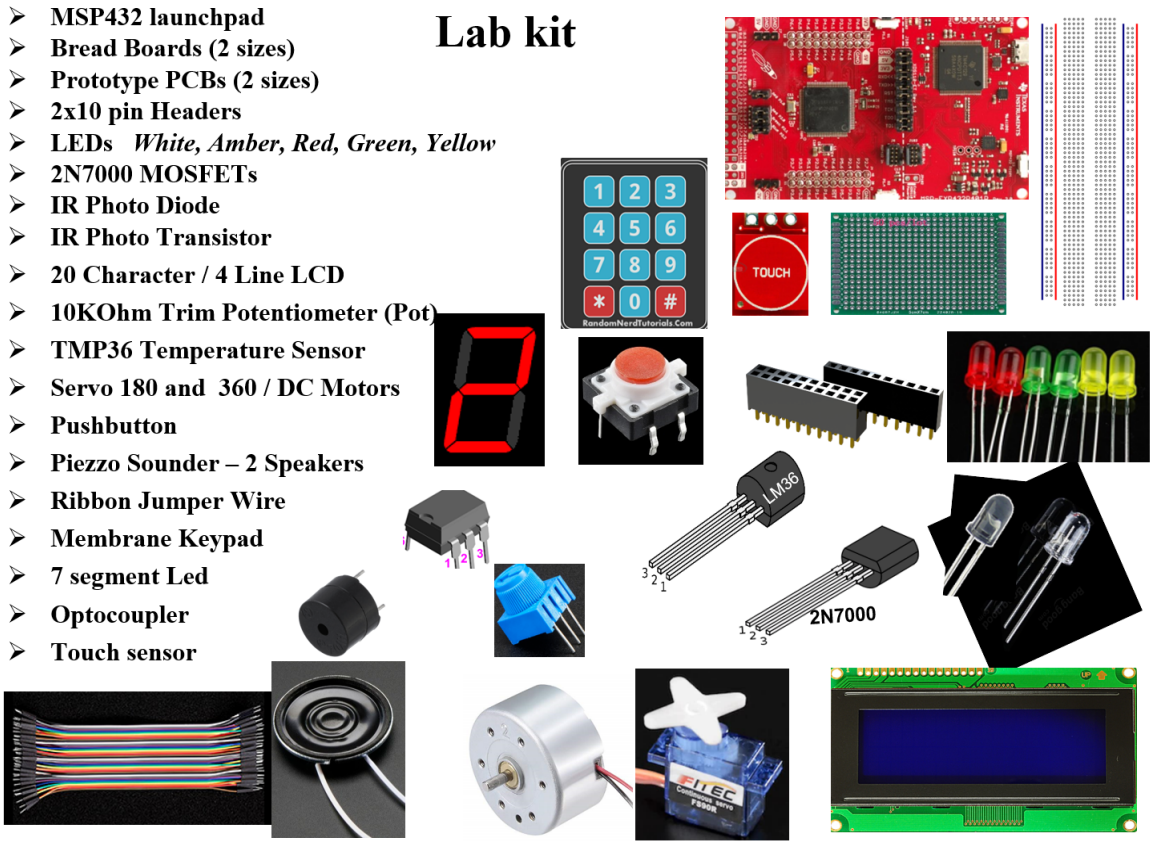

Fig. 1 Lab kit for the course

The lab kit included input devices like switches, keypad, touch sensor, photo sensor, temperature sensor and pushbutton. Output devices were LEDs, LCD screen, DC and servo motors, piezo sounder. Other circuit elements in the kit were MCU launchpad (with MCU integrated on it), breadboard, solder board, photodiode, resistors, transistors, various wires, opamp etc.

Figure 2 shows a lecture slide for a lab exercise where students learned to wire circuits for red, green and yellow LEDs and switches. And to connect them to the MCU board using transistors and resistors to limit current consumption. This slide shows a snippet of C code that was used to read the push button inputs and light LEDs based on the inputs.

Figure 3 shows the circuit used for wiring up a red LED with transistor and current limiting resistors. The value of current limiting resistors was calculated separately for green, and yellow LEDs. Snippets from this lab show how the circuits were designed and code was written to control the input output devices for various labs.

Example

1. *configure P1.1 as simple I/O in P1SEL1:P1SEL0 registers,*
2. *make P1.1 input pin in P1DIR register for push-button switch S1,*
3. *configure P1REN register to enable the pull resistor,*
4. *configure P1OUT register to select the pull-up resistor,*
5. *configure P2.0 as simple I/O in P2SEL1:P2SEL0 registers,*
6. *make P2.0 output pin in P2DIR register for red LED,*
7. *read switch from P1.1,*
8. *if P1.1(switch) is high, set P2.0 (red LED)*
9. *else clear P2.0 (red LED)*
10. *Repeat steps 7 to 9.*

```
int main(void) {
    P1->SEL1 &= ~2;                 /* configure P1.1 as simple I/O */
    P1->SEL0 &= ~2;
    P1->DIR &= ~2;                   /* P1.1 set as input */
    P1->REN |= 2;                   /* P1.1 pull resistor enabled */
    P1->OUT |= 2;                   /* Pull up/down is selected by P1->OUT */

    P2->SEL1 &= ~1;                 /* configure P2.0 as simple I/O */
    P2->SEL0 &= ~1;
    P2->DIR |= 1;                    /* P2.0 set as output pin */

      while (1) {
        if (P1->IN & 2)             /* use switch 1 to control red LED */
            P2->OUT &= ~1;          /* turn off P2.0 red LED when SW is not pressed */
        else
            P2->OUT |= 1;           /* turn on P2.0 red LED when SW is pressed */
      }
}
```

Fig. 2 Slide from a lecture showing C code for controlling LED lighting

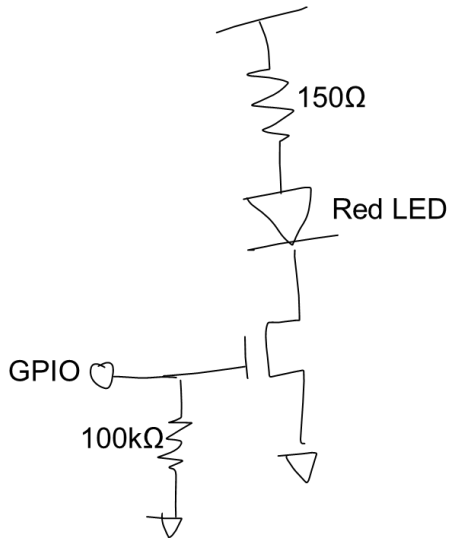

Fig. 3 Circuit diagram for wiring up the LED, transistors and current limiting resistors

c) Sample prelab question

Prelab exercises were assigned to prepare students for the lab. Students submitted their solutions online on course website. Sample solutions were posted by course staff on the course website. Exercises included circuit design problems, writing snippets of c code, especially topics that required reading before the lab. One of the problems and its solution is shown here for illustration. Students were required to solve the problem using circuit analysis principles that they learnt in prior courses.

Sample problem:

Design a proper interface to drive an LED from a port pin on the MSP432 using data sheet for the red LED from the website: www.just4funelectronics.com. (hint: assume a forward voltage about half-way up the I-V plot shown below).

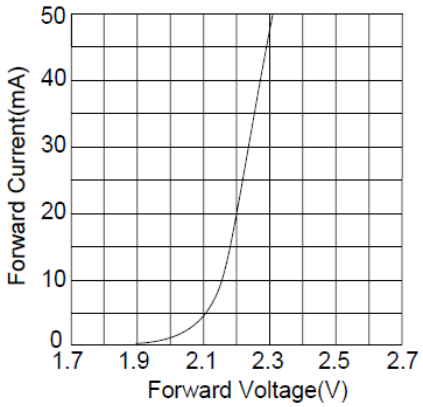

Fig. 4 IV characteristics of red LED from the data sheet.

Sample solution:

From the datasheet for the red LED, we can see that at a forward voltage of 2.2V, current of about 20mA will flow through the circuit. To enable switching of LED based on input from the MCU, the circuit in Fig. 3 includes 2N7000 MOSFET. The gate of the transistor will be connected to the input from MCU. The MOSFET is "ON" when GPIO (general purpose input output) pin is at logic high. At that time, the voltage at the bottom of the LED will be close to 0V and that on its top will be close to 2.2V. To limit the current to 20mA, across a voltage drop of 2.8V (5V – 2.2V), a current limiting resistor is required in the circuit. The value of the resistor can be calculated using the Ohm's law as

$$\mathbf{R = V / I = (5 - 2.2) / 0.020 = 2.8 / 0.020 = 140\ \Omega}$$

The closest standard resistor value that is higher than calculated value of 140 Ω is 150 Ω. Thus, the resistor value used in the circuit is 150 Ω.

d) Project

One of the end-of-term projects was to design a smart home system.

The objectives were to design a smart home and burglar alarm system using a keypad and an LCD screen in conjunction with the MCU to control LED lights, DC motor for fan, piezo sounder for burglar alarm and servo motor for a door. And to enclose the design in suitable enclosure for presentation.

Requirements/Specifications: The smart home system should allow the user to control different common home devices like fan, lights, alarm, and display temperature. User should be able to enable or disable the alarm. When the alarm is in enable state, it is triggered when the door is opened. If the alarm is disabled, it will not be triggered by opening of the door. Temperature should be read using the temperature sensor and displayed on the LCD screen. User should be able to control the fan speed and switch the fan on or off using the keypad inputs. User should be able to control the brightness of

lights and switch them on or off using keypad inputs. All the states of the devices should be displayed on the LCD screen as it is updated with keypad inputs.

In addition to integrating LCD display and keypad for output and input respectively, dc motor with fan, 3 yellow LEDs, a mechanical door controlled with the servo motor, 1 red and 1 green LED to indicate door open or close state, buzzer as alarm and the temperature sensor (read using ADC) were to be used. On powering up the system, the main menu was displayed on the LCD screen. The main menu displayed- door, alarm, lights, fan and current temperature in the room. Default state of the system was – door closed, alarm 'OFF', lights and fan 'OFF'.

First, when the alarm option was selected, the system should move to the next screen (meaning, this LCD display should clear and, another one should appear) that gives the option to either enable or disable the alarm. If the user chooses to enable the alarm, it should be triggered when the door is opened, and it should be silent when the door is closed.

Second, when door option is selected, another menu should appear to allow the user to select if they want the door open or closed. When the door is closed, a green LED should turn 'ON', and a red LED should turn 'OFF', while when the door is open, a green LED should turn 'OFF' and a red LED should turn 'ON'. If the burglar alarm is enabled, and the door is open, the alarm should also sound. Door should be opened and closed using the servo motor.

Third, when the Fan option is selected from the main menu, another menu should appear to prompt the user to enter the speed (duty cycle) they would like to run the fan at. This speed should be entered using the keypad. The fan should be run using the dc motor. An emergency stop button should be added to stop the fan immediately in case of emergency. The fan should be 'OFF' when the system powers up and is in the main menu.

Finally, when the lights option is selected from the main menu, another menu should appear in which the user can switch 3 yellow LEDs 'ON' or 'OFF'. If the user chooses to switch the lights 'ON', another screen should appear to allow the user to choose their brightness (using a PWM signal). A second button should be used to turn the light 'OFF' when pressed once. Upon a second press of the same button, the lights should regain the original color and brightness. Once the lights are turned 'ON', user should be able to switch them off using either a push button or menu option on the screen.

Other project specifications:

The LCD brightness should be controlled through a potentiometer.

The system should display the main menu after each task is completed. So, each menu had a main menu button on it.

All transitions between menus should be done using keypad. For instance, to select the alarm menu from the main menu, the user needs to press 1. Similarly, subsequent menus appearing on the LCD should have similar numbering allowing the user to select one of the options using keypad inputs. All the other inputs like duty cycle/speed should also be entered using the keypad.

Temperature need not be displayed on any menu other than the main.

One of the student projects looked like the one shown below.

Fig. 5 smart home system designed by one of the student groups

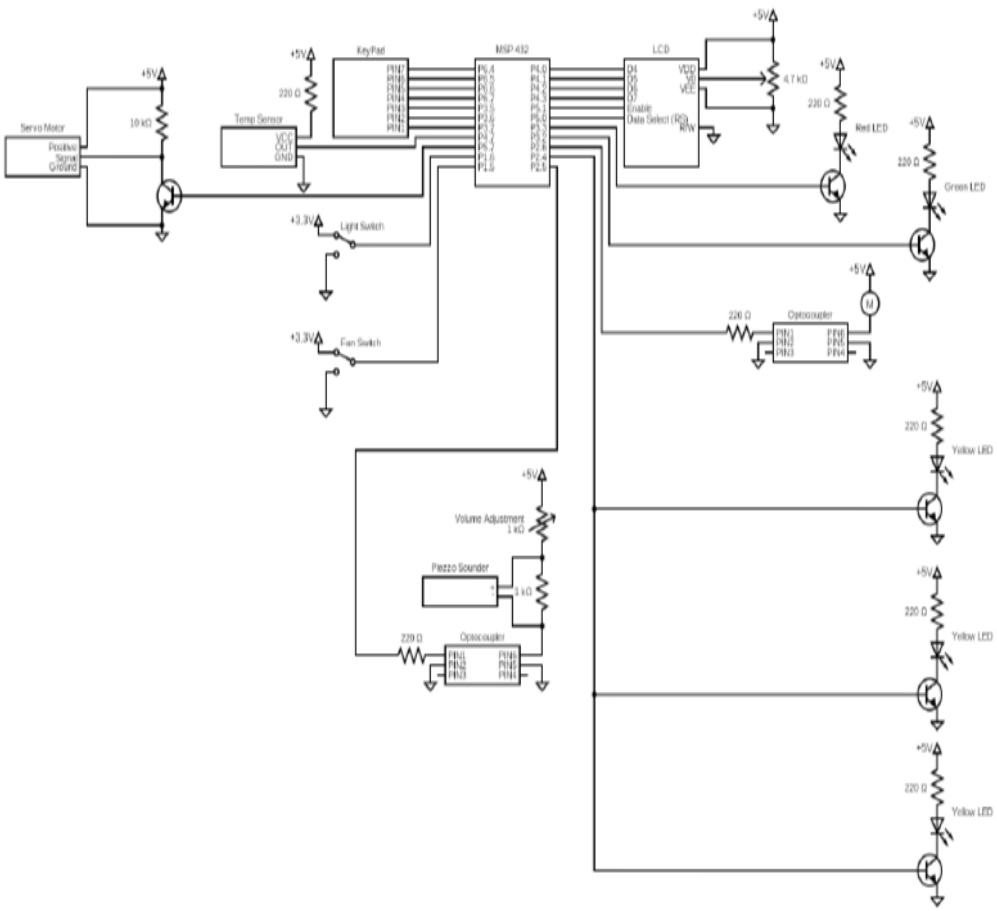

Fig. 6 Circuit for the project designed in PSPICE

The skills learnt for PCB design were used to design the circuit for the project in PSPICE as shown above. They could integrate LCD screen, keypad, motors – dc (for fan) and servo (for door), LEDs, photodiode and sensor, and piezo sounder to the MCU using the input-output ports of the MCU.

Students also gained experience in using flow charts for project design. This is exemplified in Fig. below which shows the system level design as a flow chart and indicates all the transitions.

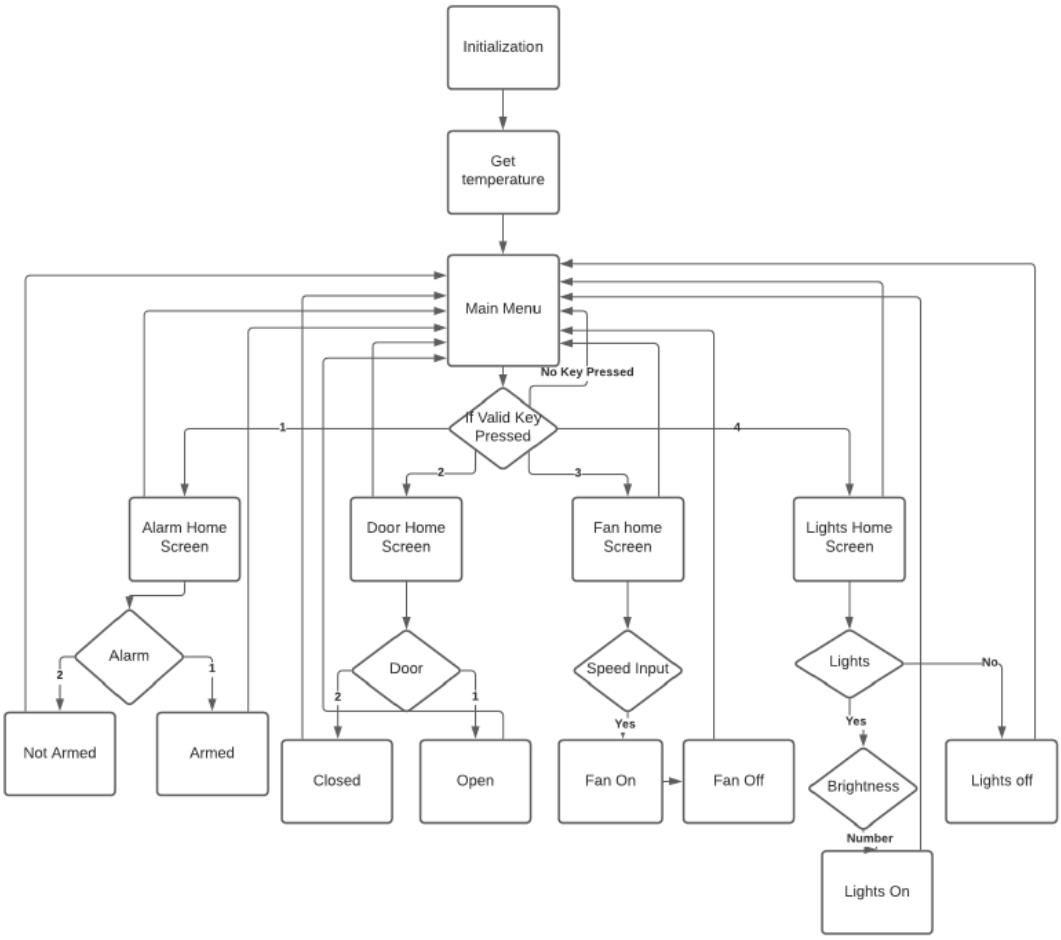

Fig. 7 Flowchart for the project

e) Reference materials

Following textbook was referred to develop course materials:

TI MSP432 ARM Programming for Embedded Systems, Mohammad Ali Mazidi, first edition [1]

In addition, the project kit and its related website [2] were referred to. For the software development, integrated development environment was downloaded from the Texas instruments website [3].

f) Credits, hours, grading

This course was offered as a 2-credit hour, elective lab course for students in Electrical and Computer Engineering. Pre-requisites included an introductory course in digital logic and analog circuits.

Class meetings included 1 one-hour lecture and 1 three-hour lab weekly. In the last 4 weeks, project-based inputs were provided, including introduction to the project, help with circuit design, coding and report writing.

Grading was based on pre-labs, labs, small projects throughout the course and two end of term projects, distributed as follows:

- *Projects: 50%*
- *Prelab: 20%*
- *Labs: 30%*

g) Course delivery: online lecture and lab

The lecture was delivered every week online via zoom. Some students attended the course from overseas. For their convenience, the lectures were recorded in zoom and recordings were made available after the lecture. To avoid zoom bombing, students had to log in to their university zoom accounts to access the zoom session.

The lab demonstration videos were posted online on course website.

Synchronous lab sessions were conducted using separate zoom session. Students worked on the labs while getting help from TA and instructor in real time. This was required because the course was offered during covid when teaching was done remotely.

4 Course outcomes

Course objectives were tested through pre-labs, labs and the end of term project (samples shown above). Performance of students on various tests, especially the project, indicated that they had absorbed all the learning objectives. They used knowledge of coding, PCB design and MCU hardware to design systems based on the MCU. And used it for applications like gaming, smart home systems etc. Upon completing the course, they have the skills to design any other system and use it for engineering or non-engineering applications. They can also integrate several VLSI chips into a system using the knowledge gained in this course.

All the students in this course performed well - 75% secured A grade and 25% secured B. None secured C or lower. In a comparable digital design course, with similar content and same instructor, 60% of the students secured A, 30% B and 10% secured C or lower grades. The difference could be because students who took this course had taken that course before and were well prepared. Only the more committed students enrolled for the course as it was an elective course with more practical, hands-on content. Students did not show interest in building neat user interface for the end of term projects leading to poor user interface design. The reason for that was felt to be little credit assigned to it. In student evaluations of the course, the rating of instructor and course content was much higher than that of comparable digital design course.

5 Student population – interest group

The students, especially those aspiring to be researchers, entrepreneurs or building MCU-based real life systems, enjoyed this hands-on course where they could build something tangible. This course was taken by 25 students in their junior and senior years of Electrical and computer engineering degree. Sophomore students in Electrical and computer engineering, students from other engineering disciplines like aerospace or mechanical and some non-engineering disciplines can also benefit from this course.

6 Conclusion & Future work

The paper describes MCU-based system design course that was offered to Electrical and Computer Engineering undergraduate students. This course is useful in several ways – firstly, students can get hands-on-experience in building systems that they have learned

about in theoretical classes. Secondly, they can use it for integrating their chip-based design projects. Thirdly, it could be very useful for future entrepreneurs. Thus, this course was a useful addition to the Electrical and Computer Engineering curriculum. This course can be offered to students with very diverse interests from every engineering and some non-engineering disciplines. In the next iteration of the course, it will be expanded to include more advanced programming concepts.

7 References

[1] Muhammad Ali Mazidi, Shujen Chen, Sepehr Naimi, "Ti Msp432 Arm Programming for Embedded Systems", 1st ed. (978-0997925913), Middletown, DE, 2016, Micro Digital Ed.

[2] Component datasheet download from <www.just4funelectronics.com>, accessed 2/9/23.

[3] CCSv10.3.1 download from www.ti.com, accessed 2/9/23.