Mini Project: Automatic Dog Feeder

By: Bryant Palomino & Kevin Coronado

Course: EE128

Demo Video:

https://drive.google.com/file/d/1WYjmuRJdxVTJumWerU8gdcYbLDjdBGRL/view?usp=sharing

Description:

The objective of this project was to design a system that would dispense food through a light source and manual input. A light source would be sensed with a photoresistor and trigger a response to the K64F from the Arduino. Other inputs would be set manually by means of a DIP switch. Our goal was to have the dispenser do one iteration in a slow and fast mode to show that it will feed a small and larger portion of food. We also wanted it to dispense after a certain period of time so we added a timer as manual input to wait and dispense after a certain period.

System Design:

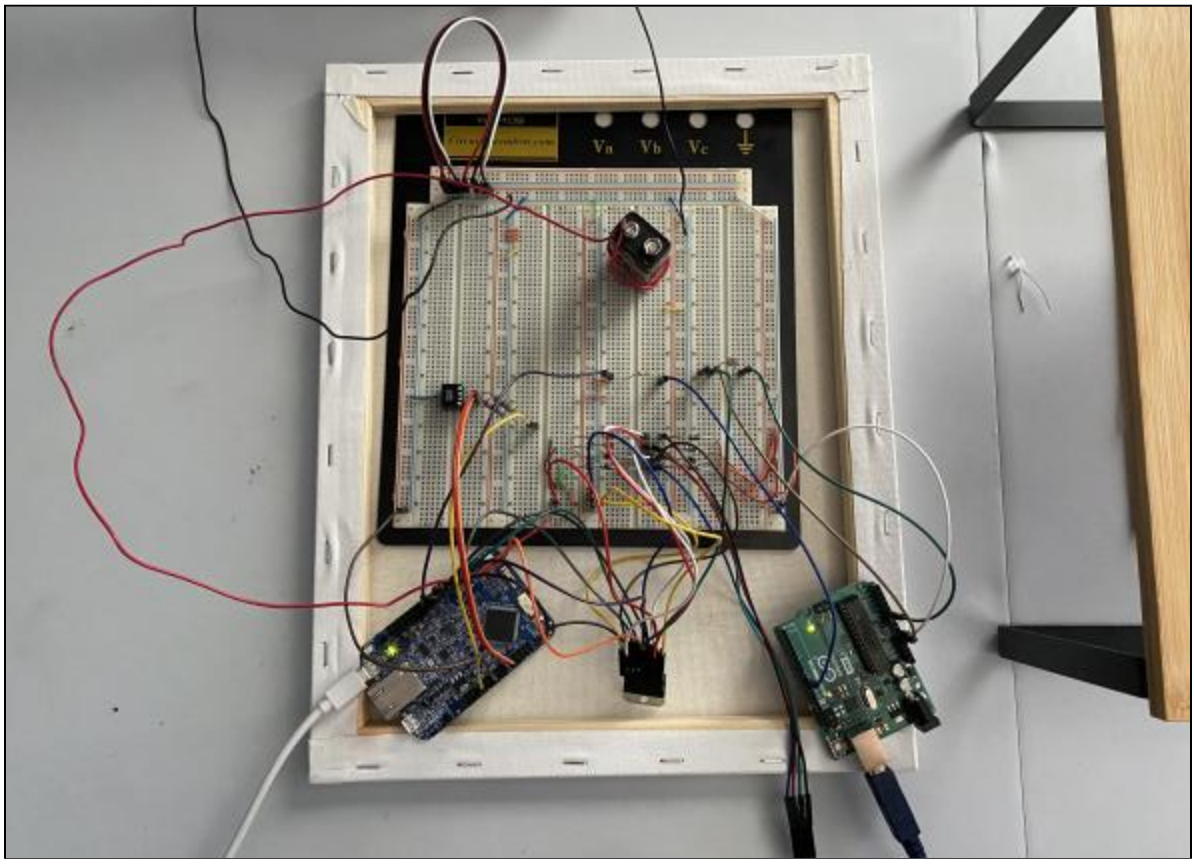

Fig 1 Physical Hardware

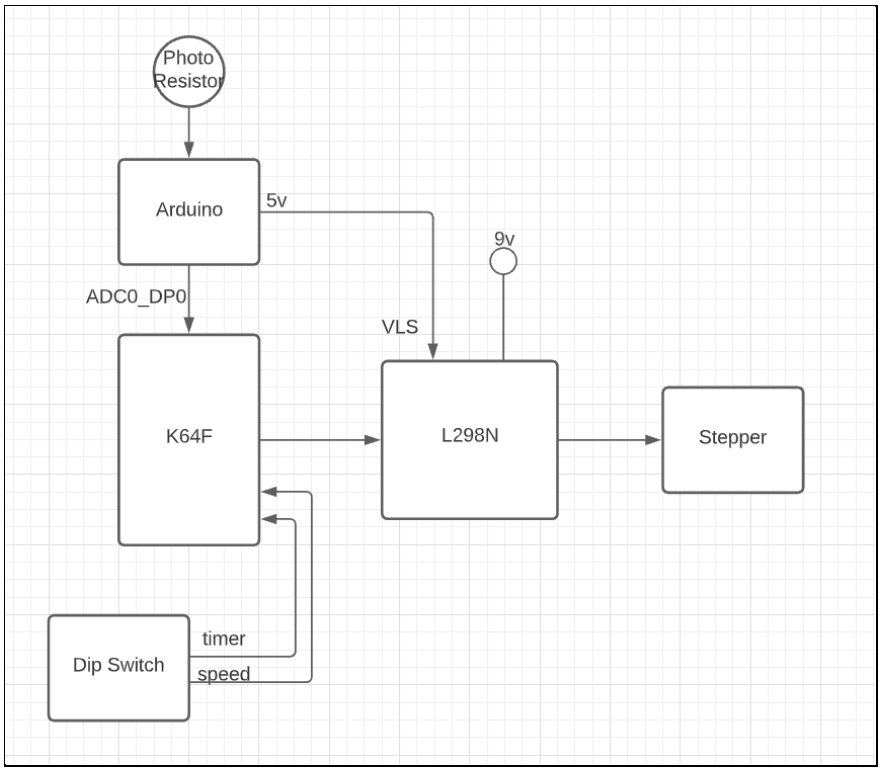

Fig 2 Hardware Implementation: Block Diagram

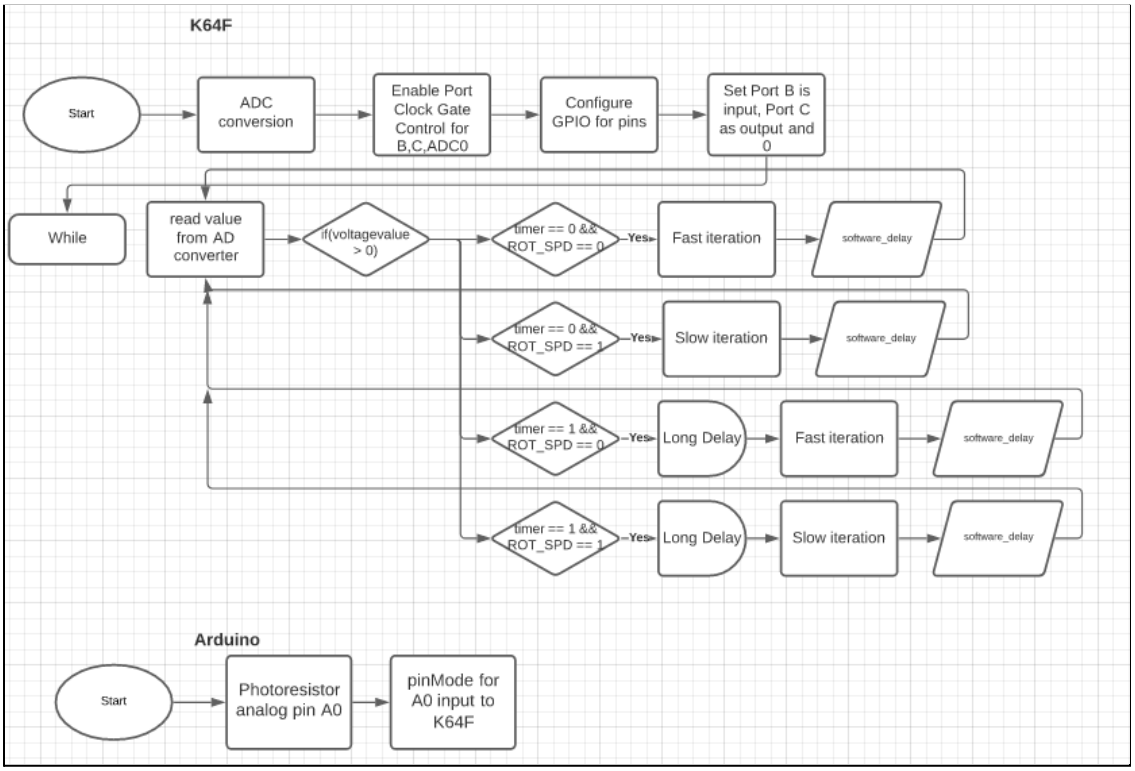

Fig 3 Software Implementation: Flowchart

Implementation Details:

Key Portion of Code:

```
if(voltagevalue > 0)
                    {
                              if (timer == 0 && ROT_SPD == 0 ){ //rotation speed fast
                                        Delay = 160000;
                                        GPIOC_PDOR = 0x36;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x35;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x39;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x3A;
                                        software_delay(Delay);
                              }
                              else if (timer == 0 && ROT_SPD == 1){ //rotation speed slow
                                        Delay = 20000;
                                        GPIOC_PDOR = 0x36;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x35;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x39;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x3A;
                                        software_delay(Delay);
                              }
                              else if (timer == 1 && ROT_SPD == 0){ // timer no delay
                                        Delay = 1000000; // Extra  delay
                                        software_delay(Delay);
                                        Delay = 160000;
                                        GPIOC_PDOR = 0x36;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x35;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x39;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x3A;
                                        software_delay(Delay);
                              }
                              else if (timer == 1 && ROT_SPD == 1){ // timer with delay set
                                        Delay = 1000000; // Extra delay
                                        software_delay(Delay);
                                        Delay = 20000;
                                        GPIOC_PDOR = 0x36;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x35;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x39;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x3A;
                                        software_delay(Delay);
                              }
```

- This portion of the code implements the fast and slow iterations of our feeder. We have 2 cases for each speed where one is for dispensing with no delay and the other with a long

delay. The cases are executed through manual input from a DIP switch with one switch being the timer and the other for speed.

Testing/Evaluation:

Test Environment:

Automatic Dog Feeder underwent testing within a brightly lit environment at room temperature. Testing was conducted within the homes of both students.

Required Equipment:

Components:	Tools:
- K64F - Arduino Uno - 9V battery - DIP switch - L298N - Stepper motor - Photoresistor	- Cylindrical container - Cardboard propeller - Cabinet shelves

Test Scenarios:

Testing scenarios were conducted under two categories, hardware and software.

Hardware tests:

1. Testing voltage/current readings, with a multimeter, across the Atmel AVR and K64F to ensure power distribution falls within the MCU parameters.
2. Measured resistance capacity of photoresistor during high and low lighting conditions.
3. Tested stepper motor response for desired output. Testing include:
    a. Ability to conduct one complete rotation without slippage.
    b. Checking various motor speeds to fit how much food exits the container.
    c. Ensuring the motor is mounted correctly when operating at desired speeds.

Software tests:

1. Tested Arduino code to output desired response

2. Debugged K64F program to ensure each input condition meetings the correct output response.

Discussions:

Challenges:

1. Actuate motor from Arduino input.
2. Ensuring we had a functioning system with the equipment used.
3. Driving the stepper motor at speeds to represent what we wanted to show.

Limitations:

1. Working with minimal equipment for hardware design.
2. Basic understanding of ADC with Arduino.

Possible Improvements:

1. Set feeder to dispense at specific times through a manual timer input.
2. Applying a sensor that measures the level of food to know when more food needs to be dispensed.
3. Use PWM instead of ADC

Roles and Responsibilities:

Bryant Palomino

- Responsible for the hardware implementation of the project. Conducted testing of the system for motor actuation and desired output response.

Kevin Coronado

- Responsible for software debugging of the project. Conducted testing on overall power delivery and circuit modeling.

Each person's roles were not mutually exclusive as both members equally shared the responsibility of constructing and testing the systems hardware and software configurations.

Conclusion:

We were able to complete the design and have our system dispense through input from the Arduino as well as input from the DIP switch. The implementation of this project was set for ADC input with Arduino sensing the presence of light through means of the photoresistor. Complications arose within how this signal was set to reach the K64F. However, the system ran correctly with the standard configuration of the ADC function within the K64F program. The desired output of food dispensing was successfully implemented and responded to each input set in place.

Appendix:

Source Code:

K64F Code

```
#include "fsl_device_registers.h"

void software_delay(unsigned long delay)
{
	while (delay > 0) delay--;
}

int ADC_read16b(void)
{
	ADC0_SC1A = 0x00; //Write to SC1A to start conversion from ADC_0
	while(ADC0_SC2 & ADC_SC2_ADACT_MASK); //Conversion in progress
	while(!(ADC0_SC1A & ADC_SC1_COCO_MASK)); //Until conversion complete
	return ADC0_RA;
}

unsigned long adcdata = 0x00;
unsigned long ADCvalue = 0x00;
unsigned long voltagevalue =0x00;

int main(void)
{
	SIM_SCGC5 |= SIM_SCGC5_PORTB_MASK; /*Enable Port B Clock Gate Control*/
	SIM_SCGC5 |= SIM_SCGC5_PORTC_MASK; /*Enable Port C Clock Gate Control*/

	SIM_SCGC6 |= SIM_SCGC6_ADC0_MASK; /*Enable ADC0 Clock Gate Control*/
	ADC0_CFG1 = 0x0C; //16bits ADC; Bus Clock
	ADC0_SC1A = 0x1F; //Disable the module, ADCH = 11111

	PORTB_GPCLR = 0x000C0100; /*Setting the pin 2 and 3 of the Port B as GPIO*/
```

```
	PORTC_GPCLR = 0x01BF0100; /*Setting the pin 0-5, 7-8 of the Port C as GPIO*/
	GPIOB_PDDR = 0x00000000; /*Setting the port B as Input*/
	GPIOC_PDDR = 0x000000FF; /*Setting the port C as Output*/
	GPIOC_PDOR = 0x00000000; /*initialize port C to 0*/
	unsigned long Delay;
	unsigned long Input=0x00;
	unsigned long timer=0x00;
	unsigned long ROT_SPD=0x00;
	while (1) {
		Input = GPIOB_PDIR & 0x0C;/*Read and Store pin 2 and 3 of the Port B*/
		timer = Input & 0x4;/*Store the pin 2 of the Port B*/
		ROT_SPD = Input & 0x8;/*Store the pin 3 of the Port B*/


		adcdata = ADC_read16b();//read value from AD converter function on Pin ADC0_DP0
		voltagevalue = (adcdata*33)/65535;   //conversion
		if (voltagevalue != voltagevalue){

			if(voltagevalue > 0 )
			{

				//lowerADC = voltagevalue/10;		//calculate the
second digital for Port C
				//GPIOC_PDOR = ADCvalueC;				//output
				//software_delay(Delay);




				if (timer == 0 && ROT_SPD == 0 ){ //rotation speed fast
					Delay = 45000;
					GPIOC_PDOR = 0x36;
					software_delay(Delay);
					GPIOC_PDOR = 0x35;
					software_delay(Delay);
					GPIOC_PDOR = 0x39;
					software_delay(Delay);
					GPIOC_PDOR = 0x3A;
					software_delay(Delay);

				}
				else if (timer == 0 && ROT_SPD == 1){ //rotation speed slow
					Delay = 20000;
					GPIOC_PDOR = 0x36;
					software_delay(Delay);
					GPIOC_PDOR = 0x35;
					software_delay(Delay);
					GPIOC_PDOR = 0x39;
```

```
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x3A;
                                        software_delay(Delay);
                                }
                                else if (timer == 1 && ROT_SPD == 0){ // timer no delay
                                        Delay = 1000000; // Extra delay
                                        software_delay(Delay);
                                        Delay = 45000;
                                        GPIOC_PDOR = 0x36;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x35;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x39;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x3A;
                                        software_delay(Delay);
                                }
                                else if (timer == 1 && ROT_SPD == 1){ // timer with delay set
                                        Delay = 1000000; // Extra delay
                                        software_delay(Delay);
                                        Delay = 20000;
                                        GPIOC_PDOR = 0x36;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x35;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x39;
                                        software_delay(Delay);
                                        GPIOC_PDOR = 0x3A;
                                        software_delay(Delay);
                                }
                        }
                }
        }
        return 0;
}
```

Arduino Code

```
const int pResistor = A0; // Photoresistor at Arduino analog pin A0
const int ledPin9=9;       // Led pin at Arduino pin 9

//Variables
int value;          // Store value from photoresistor (0-1023)

void setup(){
 pinMode(ledPin9, OUTPUT);  // Set lepPin - 9 pin as an output
 pinMode(pResistor, INPUT);// Set pResistor - A0 pin as an input
}
```