Project Demonstration: SD Calculator

Md. Shariful Islam
Student ID: 1705119
Soham Khisa
Student ID: 1705120

October 19, 2021

Contents

1 Introduction

Our simulation project SD calculator is able to:

- calculate the result of small Algebraic, Trigonometric, Inverse Trigonometric, Logarithmic and Exponential expressions up to 3 digits after decimal points,
- store the recent 9 operations,
- show the operations saved previously.

2 Description of Working Principles

2.1 Components

We have used three types of components:

1. Microcontrollers
   (a) 1 ATmega32
   (b) 1 Arduino UNO
2. Input Component
   (a) 1 Keypad ($6x4$)
   (b) 1 Keypad ($4x4$)
   (c) 2 Logical AND Gates
   (d) 1 Logical OR Gates
3. Data Management Component
   (a) LCD Display LM016L
   (b) Virtual SD Card Model with SPI Interface

Note that we did not use any sensor or actuator in our project.

2.2 Modules

The implemented modules in our project are:

- Showing expression and result in LCD display
- ATmega32 - Arduino Communication
- Storing and loading the previous operations
- Giving command to the microcontrollers

2.2.1 Showing Expression and Result in LCD Display

The LCD display is interfaced with ATmega32. From figure 1, we can see that the D2-D7 pins of LCD

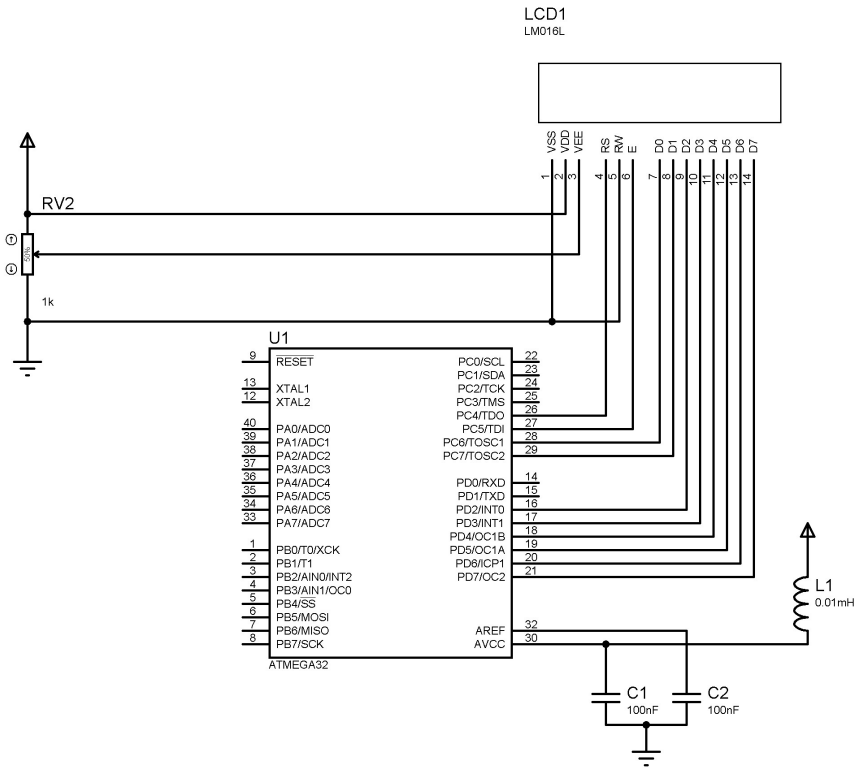

Figure 1: Interfacing LCD Display with Arduino

display are connected to PD2-PD7 of ATmega32, D0-D1 to PC6-PC7, RS to PC4, E to PC5, VSS and RW are grounded, VDD to power supple and VEE is used for contrast adjustment. The LCD display shows the expressions (typed using keypad or loaded from memory) and their results.

2.2.2 ATmega32 - Arduino Communication

When the result of an expression is calculated, ATmega32 needs to send the expression along with the result to Arduino. Besides, Arduino needs to send data to ATmega32 too when the previous operations are needed to be loaded. For this purpose, UART Serial Communication Protocol is followed as shown in figure 2.

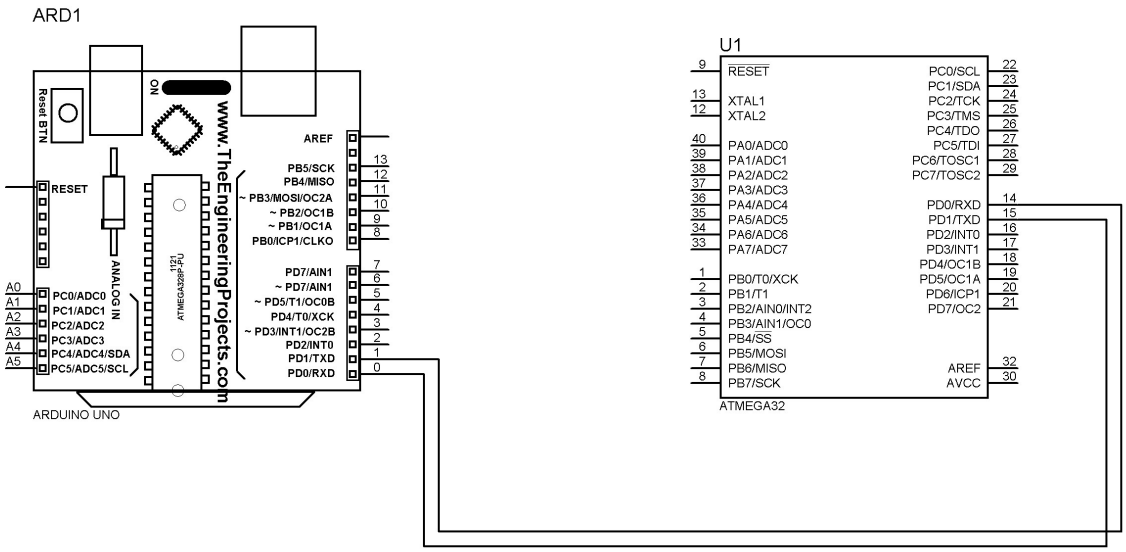

Figure 2: Arduino-ATmega32 Communication

2.2.3 Storing and Loading the Previous Operations

The SD card is interfaced with Arduino Uno to store and load data. The communication between Arduino Uno and SD card follows SPI protocol which is shown in figure 3.

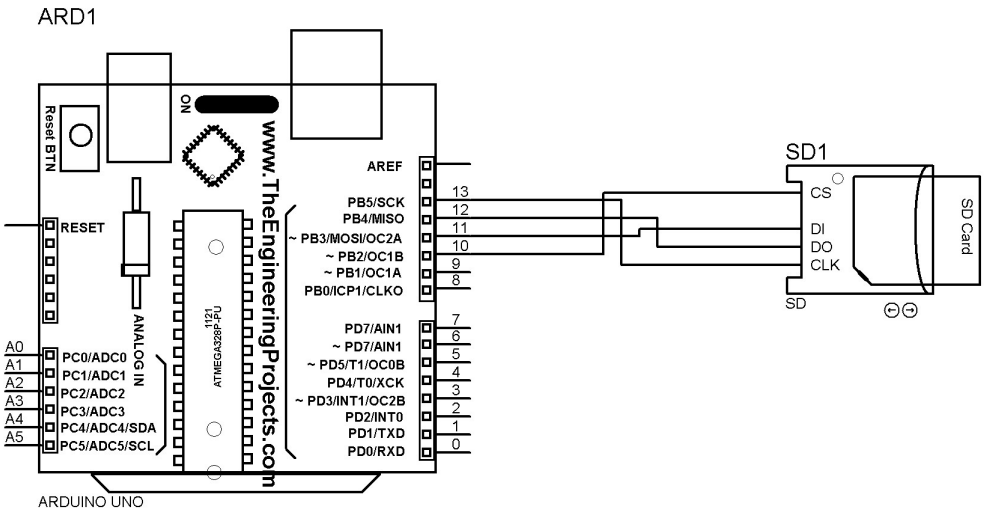

Figure 3: SD Card Interfacing

2.2.4 Giving Command to the Microcontrollers

The keypad mainly provides two types of commands - calculation command and memory command. For the purpose of calculation, the keypad is row-multiplexed by connecting the row pins to the output pins PB0, PB1, PB4, PB5 of ATmega32 and the column pins to the input pins PA0-PA7, PC2-PC3 respectively. The rows are activated periodically by making the corresponding output pin value 1 in ATmega32. When the row of a button is activated and and the button is pressed, the column value becomes 1 too. Thus having both row and column values 1 indicates that the corresponding button is pressed.

When the '=' button is pressed after typing the expression, the result is calculated and the operational data is sent to Arduino to store the data. Arduino receives data using polling mechanism.

When, "M-" button is pressed, **the interrupts INT2 of ATmega32 and INT0 of Arduino are activated during rising and falling edge** respectfully. Then data of SD Card is transferred from Arduino to ATmega32 and showed in LCD display. So does happens by pressing the "M+" button (**but the interrupt of Arduino is INT1**). "M-" is used to access the older data gradually and the "M+" button to the reverse direction. Figure 4 is given for better understanding.

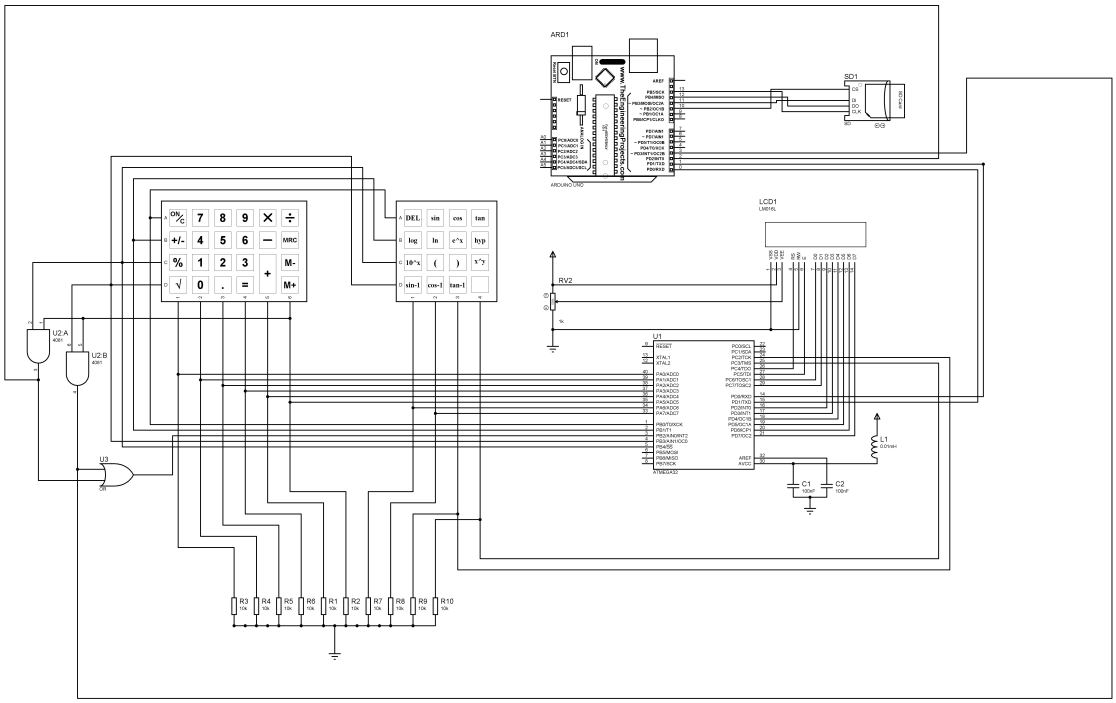

Figure 4: Circuit Diagram

3 Issues & Solutions

3.1 Virtual Terminal - A Nice Debug Tool

If any problem happens during UART serial communication, use the virtual terminal as a receiver to detect the problem. It helped us a lot.

3.2 Use interrupt when data transfer is needed

We used polling for "M+" and "M-" like other buttons of keypad at first. But it didn't work. Because to make ATmega32 receive data from Arduino, the column value must be 1 exactly at the moment when the row is active and the column value is being checked. If the button activation period ends just before the checking or begins just after the checking, data overrun is almost sure to happen. So, avoid polling as much as possible in such case.

3.3 Interrupt per task, not per button

It is not always true that different buttons need different interrupt pins. The buttons may use the same pin if the task is same. For example, we used an OR gate to use the interrupt **INT2 of ATmega32** for receiving data to check whether "M+" or "M-" button is pressed. Because in both cases, the task of Arduino is to receive operational data. Note that, we have not used the naive polling of keypad for these two buttons.

4 Conclusion

This was our first hardware project (although we have done only simulation). So, the experience was unique. Thanks to our supervisor, Md. Tareq Mahmood sir for replying spontaneously when we needed him most.