JCBAKH001 CMPJON005

EEE3096S

Prac 1

1) I2C is a bus that can be used to connect computers and embedded systems to low-speed peripherals.

2) An interrupt is a signal emitted by hardware or software to the processor that indicates something needs immediate attention. Interrupts are important because they give the user better control over the computer.

3)

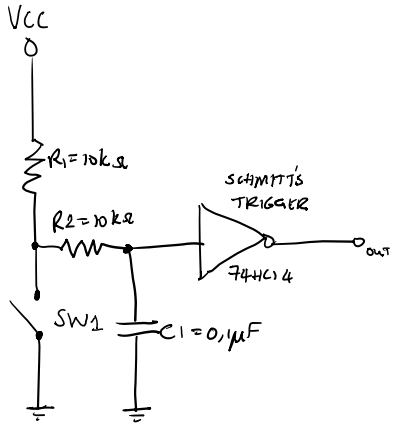
   a. Pull up and pull-down resistors make the state of the input known. A pull up resistor 'pulls' the signal to a high state and a pull-down resistor 'pulls' the signal to a low state.

   b. When a switch is pressed, there is contact bounce which can be eliminated by software or hardware debouncing. Software debouncing requires code to be written that takes the contact bounce into consideration and works around it, whereas hardware debouncing is achieved by using extra components in the circuity to eliminate the contact bounce.

Code for software debouncing:

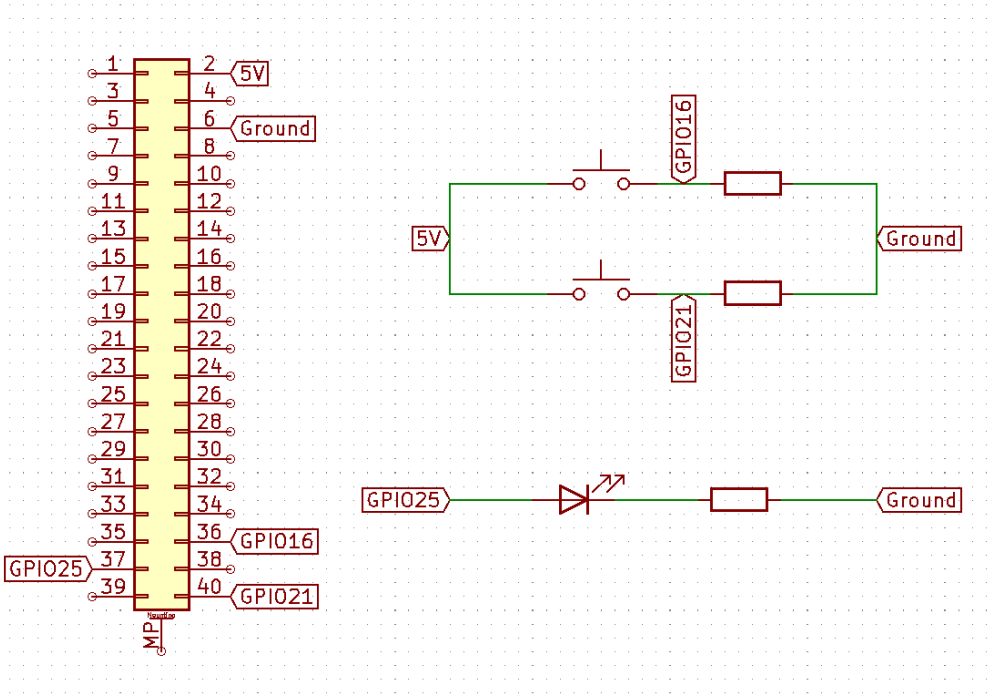
```
long interruptTime = millis();
if (interruptTime - lastInterruptTime>200){
    printf("Interrupt 1 triggered, %x\n", hours);
    . . .
]
lastInterruptTime = interruptTime;
```

Circuit diagram for hardware debouncing:



c. In ‘polling’, the CPU continuously checks the status of the devices to see if they need any attention whereas with interrupts, the device notifies the CPU that it requires attention

4. Circuit Diagram



5.

```
void hourInc(void){
        //Debounce
        long interruptTime = millis();

        if (interruptTime - lastInterruptTime>200){
                printf("Interrupt 1 triggered, %x\n", hours);
                //Fetch RTC Time
                //Increase hours by 1, ensuring not to overflow
                //Write hours back to the RTC

                hours=hexCompensation(wiringPiI2CReadReg8(RTC,HOUR_REGISTER));
                ++hours;
                hours=hFormat(hours);
                hours=decCompensation(hours);
                wiringPiI2CWriteReg8(RTC,HOUR_REGISTER, hours);

        }
        lastInterruptTime = interruptTime;
}
```

```
void minInc(void){
	long interruptTime = millis();

	if (interruptTime - lastInterruptTime>200){
		printf("Interrupt 2 triggered, %x\n", mins);
		//Fetch RTC Time
		//Increase minutes by 1, ensuring not to overflow
		//Write minutes back to the RTC
		mins=hexCompensation(wiringPiI2CReadReg8(RTC,MIN_REGISTER));
		if(mins<59){
			++mins;
			mins=decCompensation(mins);
			wiringPiI2CWriteReg8(RTC,MIN_REGISTER,mins);
		}
		else{
			wiringPiI2CWriteReg8(RTC,MIN_REGISTER,0);
			hourInc();
		}

	}
	lastInterruptTime = interruptTime;
}
```

6.

https://github.com/Jonathan5320/EEE3099S-.git