BLINKY WITH CONSOLE

ASSIGNMET – 03

BASATI SIVAKRISHNA

22976

DESE EPD

Code explanation:

UART and GPIO (pushbuttons) has configured for the interrupt actions. The handlers have implemented in startup file for both interrupts. Upon receiving the interrupts the handlers will clear the received interrupt flag and set the variables values accordingly.

INTERRUPT HANDLER(UART):

```
void UARTIntHandler(void)
{
   uint32_t stat;
   stat = UARTIntStatus(UART0_BASE, true);
   UARTIntClear(UART0_BASE, stat);
   while(UARTCharsAvail(UART0_BASE))
   {
       rx_val[command_index] = UARTCharGetNonBlocking(UART0_BASE);
       if(rx_val[command_index]== '\n')
       {
           command_rxd = 1;
       }
//       UARTCharPutNonBlocking(UART0_BASE, rx_val[i]);
       (command_index>30)? (command_index = 0): (command_index++);
   }

}
```

The handler written in such a way that only 30 characters will be received and character after 30th will be kept at position 1 of rx_val[] array.

Once new line(\n) is encountered the entire data received via UART will be passed to command parser () function.

Command parser will get modified accordingly (removing extra characters and capital letters). The modified command will get divided into two parts. 1. Parsed_cmnd and 2. Parsed data.

And it will check the validity command and data using ascii values.

Whenever there is an invalid command, the command_parser() function will set the valid flag to 0 and it prints the command help message.

Logic to reject the unnecessary characters in received buffer.

```
while((rx_val[a] != '\n'))
{
        if(!(((rx_val[a]>47) && (rx_val[a]<58)) || ((rx_val[a]>64) && (rx_val[a]<91)) || ((rx_val[a]>96) && (rx_val[a]<123))))
            {
            a++;
            continue;
            }
        else
        {
            if((rx_val[a]>64) && (rx_val[a]<91))
                rx_val[a] = rx_val[a] + 32;
            modify_cmnd[b] = rx_val[a];
            a++;
            b++;
        }
}
```

If the command given is valid then according to the given command either respective led blinks or blink rate changes.

A switch case structure is used to determine the type of led to glow. and small mathematical formula is used to determine blink rate based on the command.

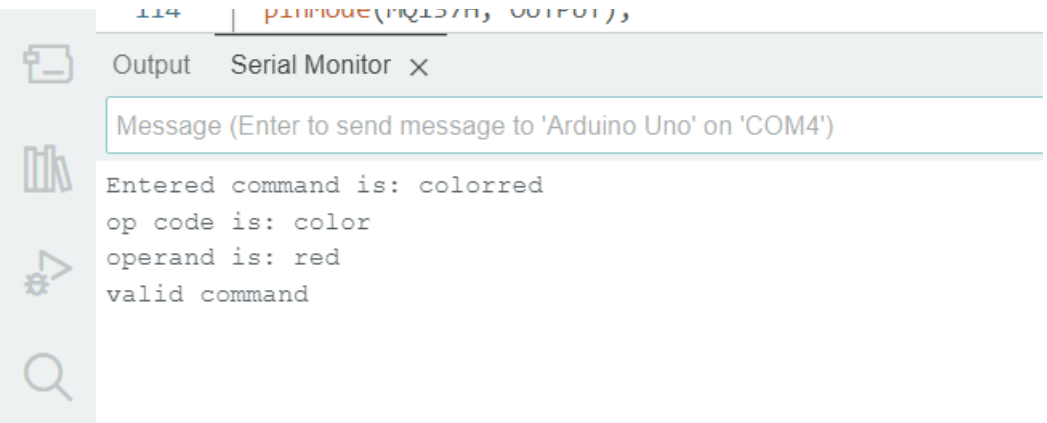
blink blink_rate: Delay calculation.

```

        }
        else if((strcmp(parsed_cmnd, "blink")==0))
        {
            counter = atoi(parsed_data);
            counter = ((60000)/(2*counter));
            cmd_mode = 1;
        }
        }
```

OUTPUT: <u>Modified command will be echoed without spaces</u>

Using valid command:



Using invalid command:

```
Output    Serial Monitor ×
Message (Enter to send message to 'Arduino Uno' on 'COM4')
Entered command is: colorred
op code is: color
operand is: red
valid commandEntered command is: blankred
op code is: blank
operand is: red
**********  ENTERED COMMAND IS INVALID  *******
ENTER THE FOLLOWING COMMANDS ONLY
1. blink blink_rate        2. color color_type
```

Entering the extra characters like special characters and spaces.

```
Output    Serial Monitor ×
bl & i n!k 1@20
```

```
Output    Serial Monitor ×
Message (Enter to send message to 'Arduino
Entered command is: blink120
op code is: blink
operand is: 120
valid command
```