CONESTOGA
Connect Life and Learning

EECE8040

ENGINEERING CAPESTONE PROJECT

ACHIEVERS

GROUP D – THE ACHIEVERS

ACCESS CONTROL AND RECORDING

Initial Software Creation

Supervisor Name

Prof. Sandra French

Team Members

Sanket Pancholi

Bhumi Patel

Muhammad Sumair Dagiya

Index

SR NO	Content	Page
1	Introduction	03
2	Peripheral interface	03-04
2.1	MFRC 522 RFID reader	03
2.2	PC interface	03
2.3	LCD	04
2.4	Switches	04
3	Pin configuration	04
4	Source code	05-09
4.1	SPI initialization	05
4.2	Timer initialization	06
4.3	USART initialization	07
4.4	GPIO initialization	08
4.5	Result	09
5	Flowchart	10-11
5.1	Entry door	10
5.2	Exit door	11
6	GitHub creation	12

1. Introduction

For Access control and recording system, we are using many component so we need to interface them with microcontroller STM32F303.Here in this document we have mentioned all the peripheral interfaces and source code for their initialization. Project flow chart is included in third section.

2. Peripheral Interface

The peripherals which we are going to use in this projects are as follows:

- MFRC 522 RFID Reader
- PC Interface
- LCD
- Switches and LED

So for that initially we have interfaced the pins from microcontroller to that peripherals.

2.1. MFRC 522 RFID Reader

Function	MRFC 522 RFID READER PIN NO	STM32 Nucleo Board PIN NO
SDA	1	CN10-17
SCK	2	CN10-11
MOSI	3	CN10-15
MISO	4	CN10-13
IRQ	5	NC
GND	6	CN10-9
RST	7	CN10-19
3.3V	8	CN7-16

Table 2.1- RFID interface

2.2. PC Interface

Function	STM32 Nucleo Board PIN NO
TX	CN7-1
RX	CN7-2

Table 2.2- Computer interface

2.3. LCD

Function	STM32 Nucleo Board PIN NO
RS	CN7-38
R/W	CN7-36
E	CN7-35
D0	NC
D1	NC
D2	NC
D3	NC
D4	CN7-37
D5	CN10-34
D6	CN10-6
D7	CN10-4

Table 2.3- 16x2 LCD interface

2.4. Switches

Switch NO	STM32 Nucleo Board PIN NO
SW0	CN10-16
SW1	CN10-30
SW2	CN10-28

Table 2.4- Switch interface

3. Pin configuration

After allocating all this pin we have started making code and for that we have done pin configuration in Atolic Trustudio.

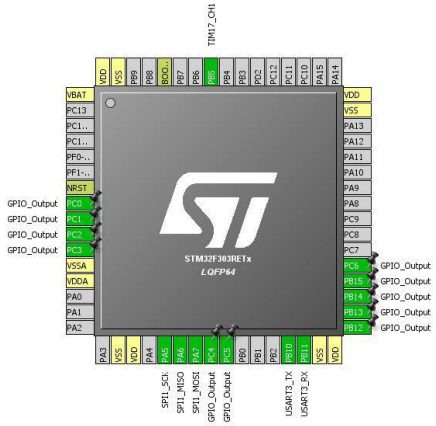

Fig 3.1- Pin allocation

4. Source code

Here we have included source code for peripheral initialization.

4.1. SPI initialization

```
static void MX_SPI1_Init(void){
/* SPI1 parameter configuration*/
hspi1.Instance = SPI1;
hspi1.Init.Mode = SPI_MODE_MASTER;
hspi1.Init.Direction = SPI_DIRECTION_2LINES;
hspi1.Init.DataSize = SPI_DATASIZE_4BIT;
hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
hspi1.Init.NSS = SPI_NSS_SOFT;
hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
hspi1.Init.CRCPolynomial = 7;
hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
hspi1.Init.NSSPMode = SPI_NSS_PULSE_ENABLE;
if (HAL_SPI_Init(&hspi1) != HAL_OK)
{
_Error_Handler(__FILE__, __LINE__);
}
}
```

4.2. Timer initialization

```
static void MX_TIM17_Init(void)
{
  TIM_OC_InitTypeDef sConfigOC;
  TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig;
  htim17.Instance = TIM17;
  htim17.Init.Prescaler = 0;
  htim17.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim17.Init.Period = 0;
  htim17.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim17.Init.RepetitionCounter = 0;
  htim17.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
  if (HAL_TIM_Base_Init(&htim17) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
  if (HAL_TIM_OC_Init(&htim17) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
  sConfigOC.OCMode = TIM_OCMODE_TIMING;
  sConfigOC.Pulse = 0;
  sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
  sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
  sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
  sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
  sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
```

```
  if (HAL_TIM_OC_ConfigChannel(&htim17, &sConfigOC, TIM_CHANNEL_1) !=
HAL_OK)
{
   _Error_Handler(__FILE__, __LINE__);
  }
  sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
  sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
  sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
  sBreakDeadTimeConfig.DeadTime = 0;
  sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
  sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
  sBreakDeadTimeConfig.BreakFilter = 0;
  sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
  if (HAL_TIMEx_ConfigBreakDeadTime(&htim17, &sBreakDeadTimeConfig) !=
HAL_OK)
  {
   _Error_Handler(__FILE__, __LINE__);
  }
  HAL_TIM_MspPostInit(&htim17);
}
```

4.3. USART initialization

```
static void MX_USART3_UART_Init(void)
{
  huart3.Instance = USART3;
  huart3.Init.BaudRate = 38400;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
```

```
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }
}
```

4.4. GPIO initialization

```
/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct;
  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                    |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6, GPIO_PIN_RESET);
```

```
  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB,
GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);
  /*Configure GPIO pins : PC0 PC1 PC2 PC3
                        PC4 PC5 PC6 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                      |GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
  /*Configure GPIO pins : PB12 PB13 PB14 PB15 */
  GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
}
```

4.5. Result

After initializing all these we checked the led with interfacing that to an output pin.

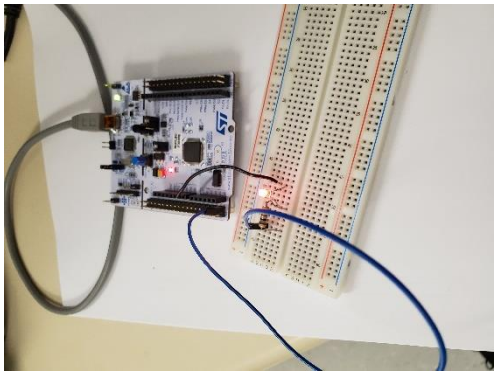

Fig 4.5.1-Result

5. Flowchart

5.1. Entry Door

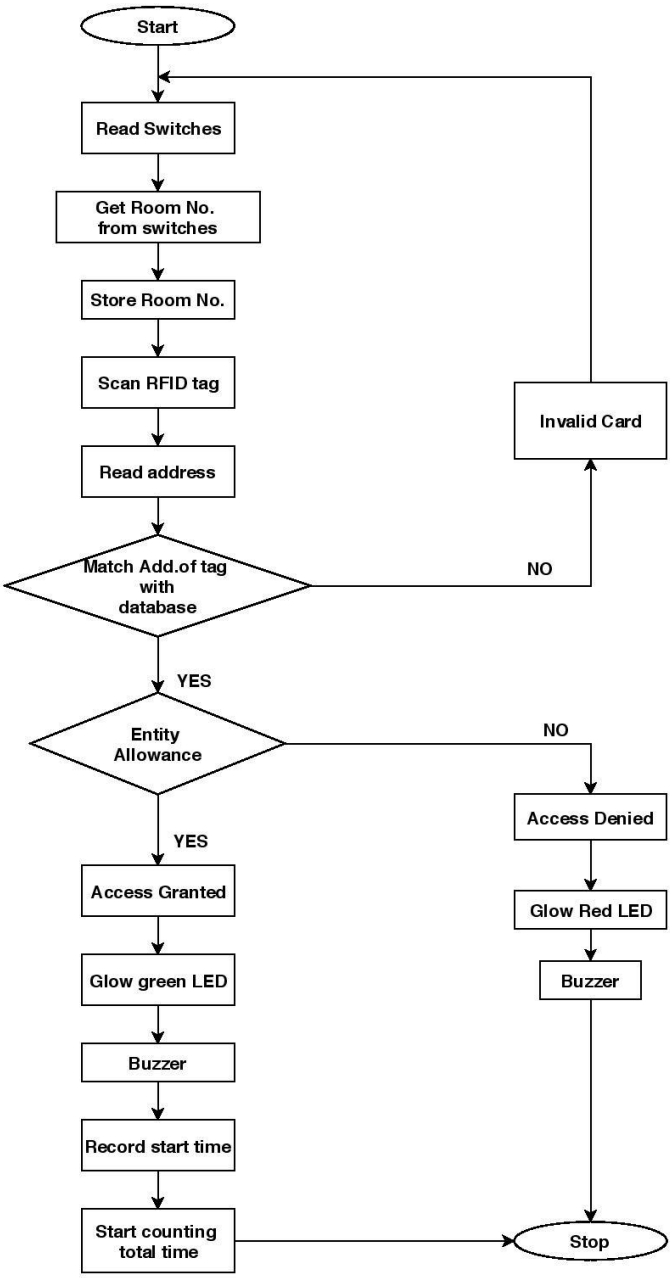

Fig 5.1.1- Flowchart

5.2. Exit Door

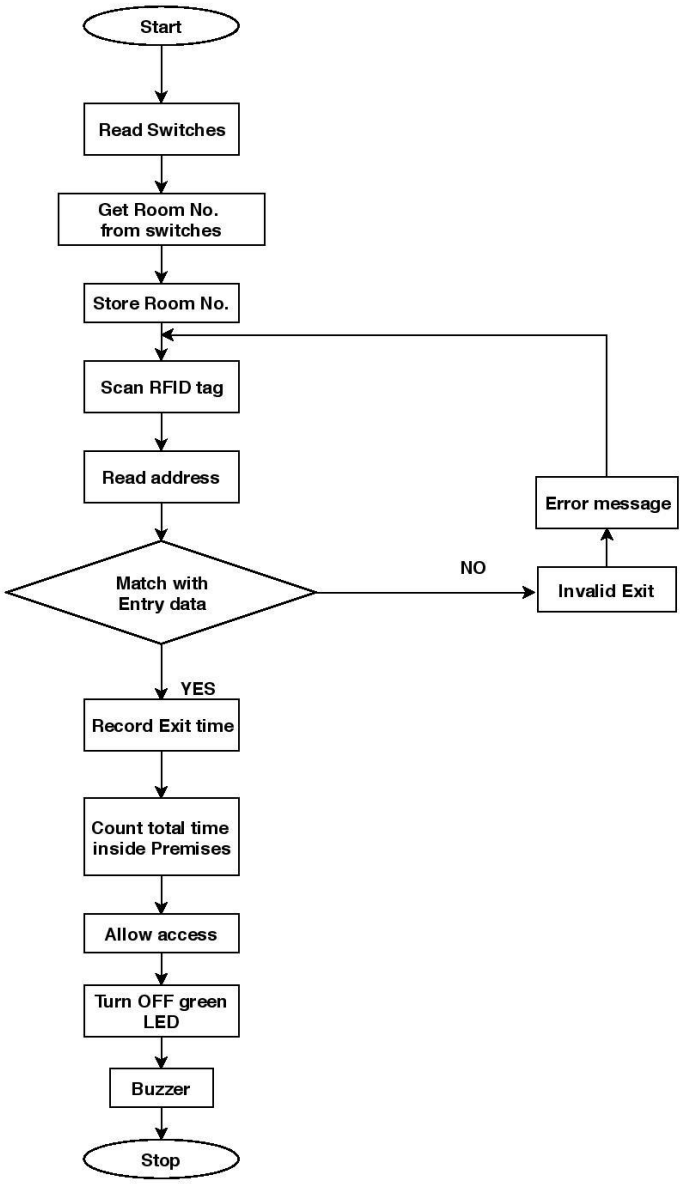

Fig 5.2.1- Flowchart

6. GitHub Creation

- We have started with creating GitHub account and made a repository.

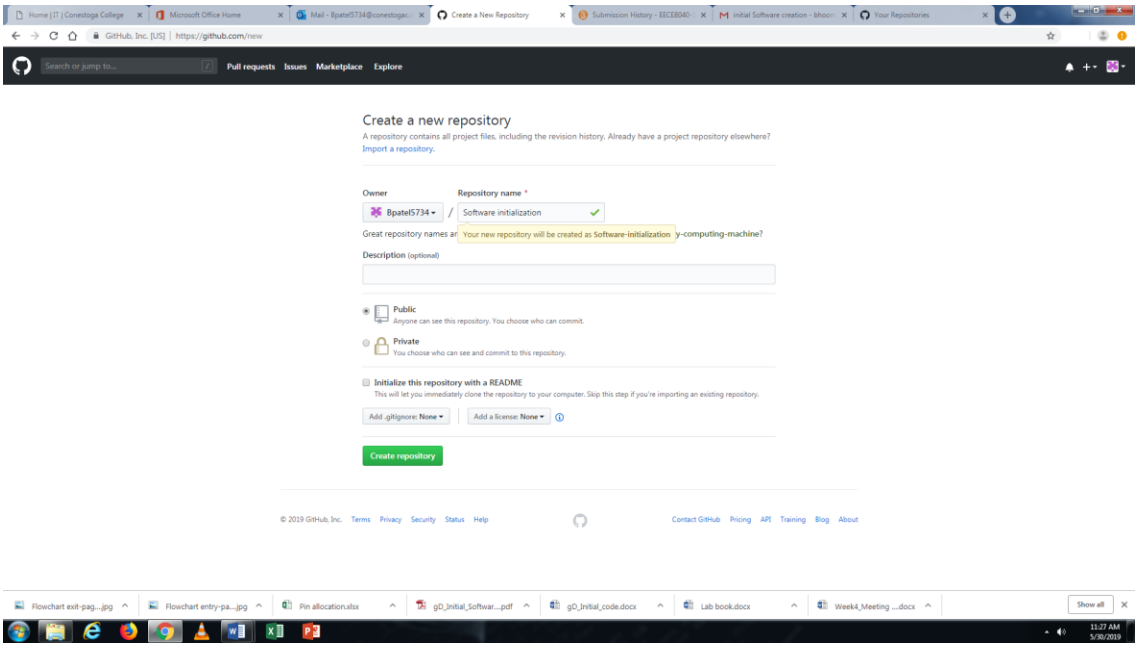

Fig 6.1- Creating a repository

- **GitHub Collaboration**

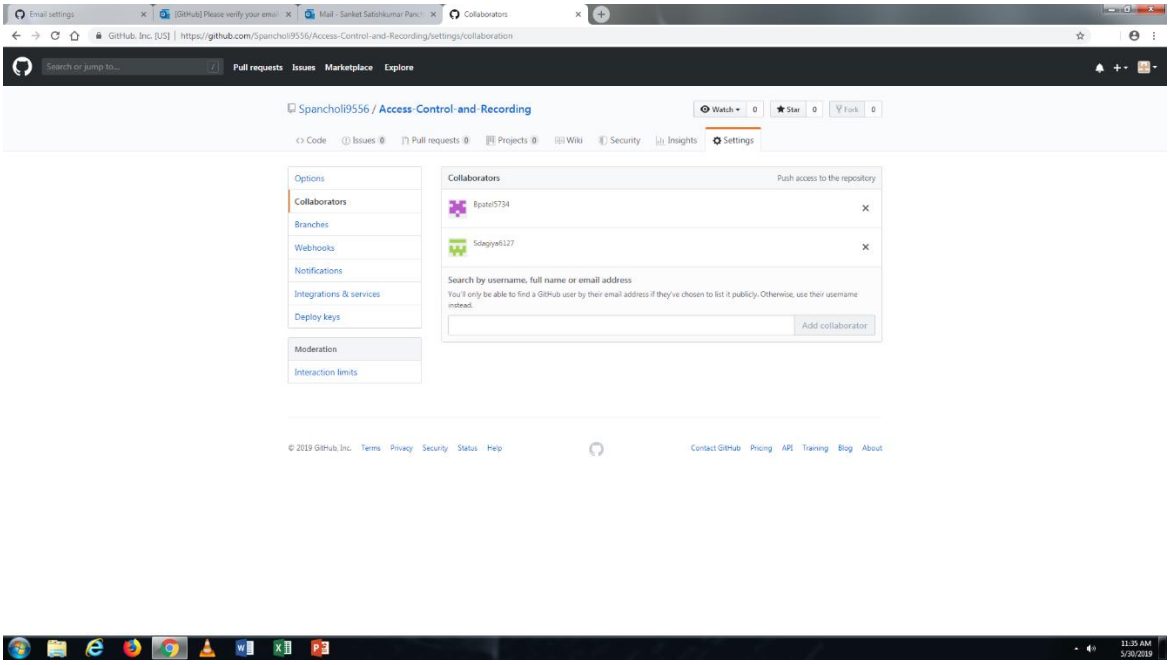

Fig 6.2- GitHub Collaboration

- **GitHub account links**
  - https://github.com/Bpatel5734/Initial-software-creation
  - https://github.com/Sdagiya6127
  - https://github.com/Spancholi9556