Report 1

Alex Hu, Shawn Hsiao, Shih-Hao Yeh

ABSTRACT:
In this project, we did several things to practice out programming in C. First, we wrote C code on CodeBlock to perform different display task on terminal (with some delay function). The result delay time is not exactly as expected if using for loop. However, we still can get an approximate time delay. And then, we debugged the two C codes provided in the next section and get to know more about some common error that frequently happen while programming in C. At the later part, we utilized the display and delay function and implemented onto the Arduino using TFT. The result is almost the same as in using terminal. On the other hand, we tested out the serial communication between Arduino Uno and ATMega. The result shows that the communication between devices is built successfully.

INTRODUCTION:
C is a fundamental programming language with many features, such as pointer, casting, and so on. With all these features, we can build up a system that is efficient in time and memory. However, if not using well, the system might crash easily. For example, if a pointer point to the address that is not intended to be point to, then you might change some variables that are not supposed to be changed. Because of these situations, it's worthwhile practicing out programming skill in C before starting to build up a larger embedded system.

In this project, before implemented code onto the Arduino, we first tested out C code using our terminal. We are asked to display different letters (A, B, C, D) in 8 different kinds of way.

In the next part, we were provided with 2 C code and are asked to debug the two file. Both of them can be compiled without any error message, but the second one will have error when executed. Our goal is to find out where are the mistakes, and how to correct them.

After debugging the code, we utilized the display function and delay function derived in the first step and implemented onto the Arduino. This time we are using the TFT display instead of the computer terminal, so we also need to know the function call for displaying on TFT.

At the end, we test the serial communication between Arduino Uno and ATMega. Communication between devices need some setup steps done before connecting, such as baud rate and TX RX port.

DESIGN SPECIFICATION:

Programming in C on CodeBlock

- Application 1: Display the letters 'A'. 'B', 'C' and 'D' in console and flash them together at approximately a one-second rate.
- Application 2: Modify the program in Application 1 to print then erase the letters A then B then C then D at approximately a one-second rate.
- Application 3: Modify the program in Application 1 to flash the letters A and C at a one-second rate and the letters B and D at a two-second rate.
- Application 4: Modify the program to parameterize the two delay() function calls in the two for loops so that they will support different user specified delays rather than the single hard coded value.
- Application 5: Modify the program so that each of the two respective for loops is replaced by the following functions; void f1Data(unsigned long delay1) and void f2Clear(unsigned long delay2).
- Application 6: Modify the program so that parameter, delay, is passed into the two functions in Application 4 above by reference rather than by value.
- Application 7: Modify the program so that the two functions in Application 5 above are replaced by a single function. The function should be able to be called with the character to be displayed and the value of the delay.
- Application 8: Modify the program so that the function in Application 6 is in a separate file.

Use for loop as delay function and test the function empirically.

Debug of given C codes

- Correct the error shown during compilation
- Correct the possible error in code although not causing an error in compilation

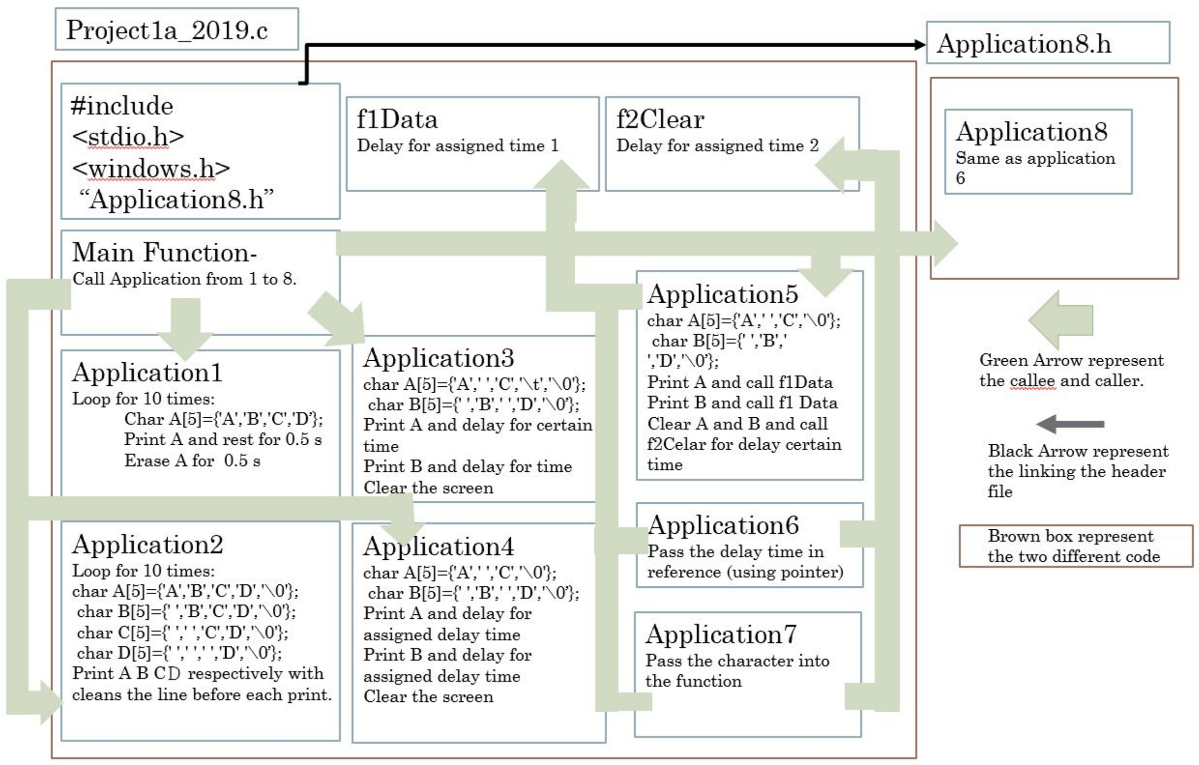
Programming in Arduino IDE

- Setup ATMega board to use the TFT display
- Port Application 7 from CodeBlock onto ATMega board.
- Display the result using TFT.

Arduino Serial Communication

- Wire TX1, RX1 on ATMega board with RX, TX on Uno board.
- Setup serial communication between ATMega board and Uno board.
- Program ATMega board to send a string and output the received string in serial monitor.
- Program Uno board to send the received string back.

SOFTWARE IMPLEMENTATION:


Project1a_2019.c
Application8.h
#include <stdio.h> <windows.h> "Application8.h"
f1Data
Delay for assigned time 1
f2Clear
Delay for assigned time 2
Application8
Same as application 6
Main Function-
Call Application from 1 to 8.
Application5
char A[5]={'A',' ','C','\0'}; char B[5]={' ','B',' ','D','\0'};
Print A and call f1Data
Print B and call f1 Data
Clear A and B and call f2Celar for delay certain time
Application1
Loop for 10 times:
Char A[5]={'A','B','C','D'};
Print A and rest for 0.5 s
Erase A for 0.5 s
Application3
char A[5]={'A',' ','C','\t','\0'}; char B[5]={' ','B',' ','D','\0'};
Print A and delay for certain time
Print B and delay for time
Clear the screen
Green Arrow represent the callee and caller.
Black Arrow represent the linking the header file
Brown box represent the two different code
Application6
Pass the delay time in reference (using pointer)
Application2
Loop for 10 times:
char A[5]={'A','B','C','D','\0'};
char B[5]={' ','B','C','D','\0'};
char C[5]={' ',' ','C','D','\0'};
char D[5]={' ',' ',' ','D','\0'};
Print A B CD respectively with cleans the line before each print.
Application4
char A[5]={'A',' ','C','\0'}; char B[5]={' ','B',' ','D','\0'};
Print A and delay for assigned delay time
Print B and delay for assigned delay time
Clear the screen
Application7
Pass the character into the function

In the project1a_2019.c, the there is 9 function including the main function. The green arrow shows that the control flow of the program. For example, the main function will call each of Application from 1 to 8. And for Application 5 to 8, 4 of them will call the f1Data and f2Clear inside their function. For each application pseudo code, the viewer can refer to the diagram shown above. In addiction, the application 8 is stored in the application.h header file which is #include by project1a_2019.c in the very beginning and it satisfied the requirement of application 8.

TEST PLAN:

The correctness of output both in console and on TFT is tested. In addition, we will make sure the given codes are bug free. Last, the serial communication between ATMega board and Uno board is tested.

TEST SPECIFICATION:

1. The output of C code in console and on TFT should corresponds to the input parameters of the function in code.
2. Provided codes should be bug free.

3. The received message from the serial communication should be the same as the message sent.

TEST CASES:

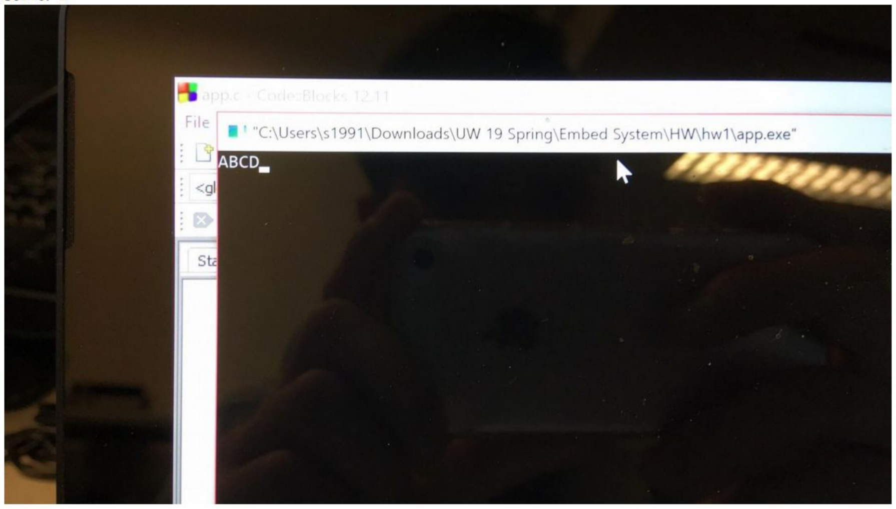
1. Observe from the console and TFT to see if the letters A and C are flashing at a one-second rate and the letters B and D are flashing at a two-second rate.
2. Compile the given codes and check for any error.
3. Send letters 'A', 'B' and 'C' sequentially and check if the serial monitor of Uno board is showing the same letters.

RESULT ANALYSIS:

The result of application from screen shot and picture of implementation of TFT on Arduino are shown below.

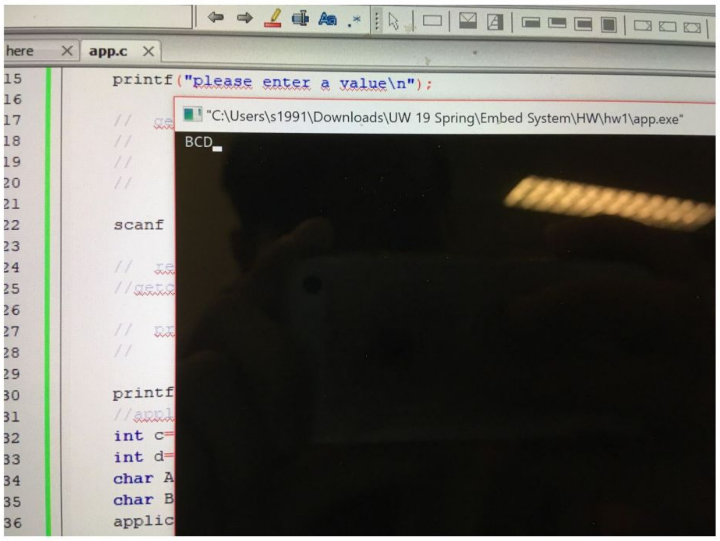
Display (First Part)

Part 1.
(1) Applicant. 1



```
"C:\Users\s1991\Downloads\UW 19 Spring\Embed System\HW\hw1\app.exe"
ABCD_
```

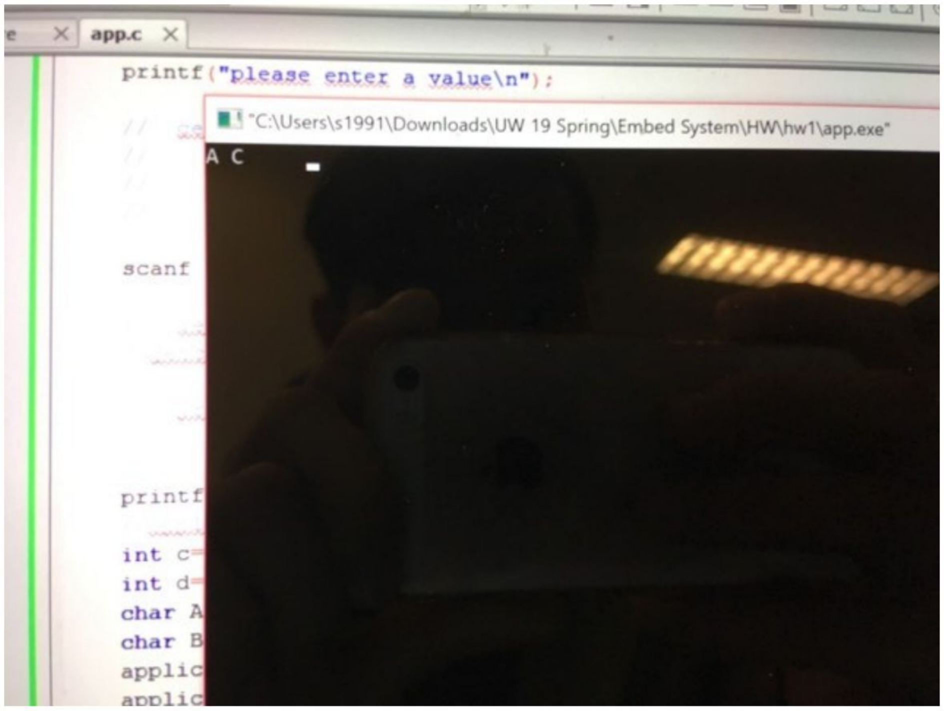
(2) Applicant. 2



```
printf("please enter a value\n");

//  ge
//
//
//

scanf

//  re
//getc

//  pr
//

printf
//appl
int c=
int d=
char A
char B
applic
```

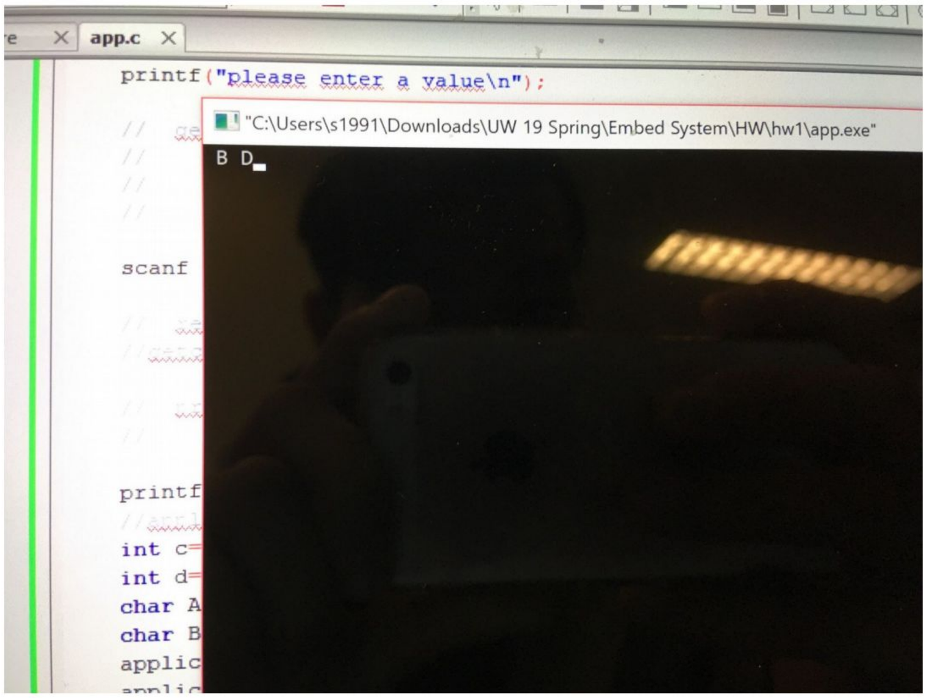
```
"C:\Users\s1991\Downloads\UW 19 Spring\Embed System\HW\hw1\app.exe"
BCD
```

(3) Applicant. 3, 4, 5, 6, 7, 8



```
printf("please enter a value\n");

//  ge

scanf

printf

int c=
int d=
char A
char B
applic
applic
```

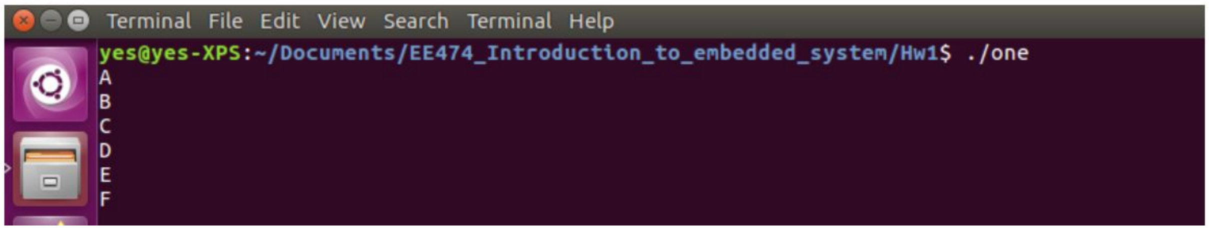
```
"C:\Users\s1991\Downloads\UW 19 Spring\Embed System\HW\hw1\app.exe"
A C
```



```
printf("please enter a value\n");
//
//
//
//
scanf
//
//
//
//
printf
//
int c=
int d=
char A
char B
applic
```

```
"C:\Users\s1991\Downloads\UW 19 Spring\Embed System\HW\hw1\app.exe"
B D_
```

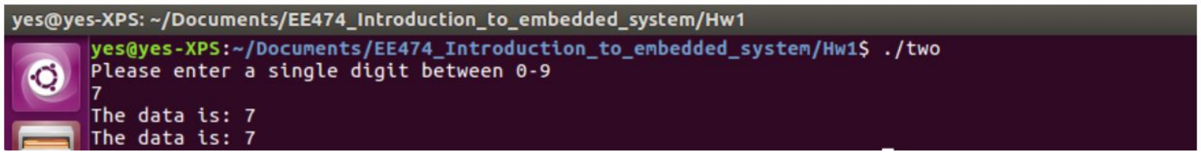
Debugging (Second Part)
In the first code (project1_2019b), the bug is related to array out of range. The reason why it can still execute is that the address do exist. Therefore, you can still access it. However, if the address is take up by other variable, we might get unexpected result.



```
yes@yes-XPS:~/Documents/EE474_Introduction_to_embedded_system/Hw1$ ./one
A
B
C
D
E
F
```

* Solution: We simply change the definition of myArray to 6 element, which is char myArray[6]. Therefore, the sixth space is reserved for myArray.

In the second code (project1_2019c), the bug is related to local address being deleted after the function is finished. If a pointer is pointed to a local variable address, after the function finished, the address will not contain anything, and we will not get the result we expect that is in the function.

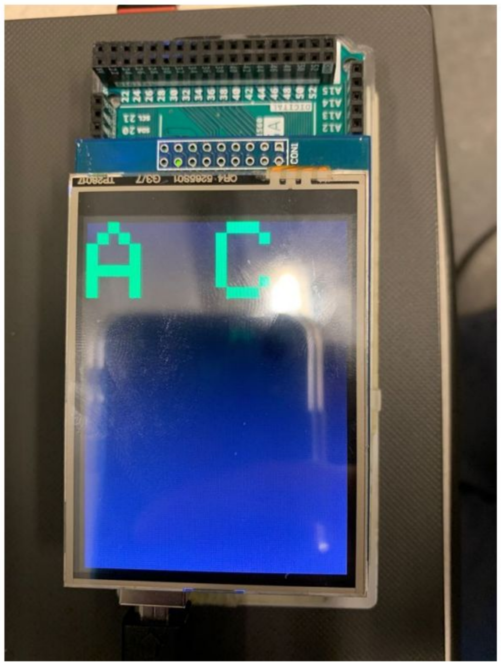


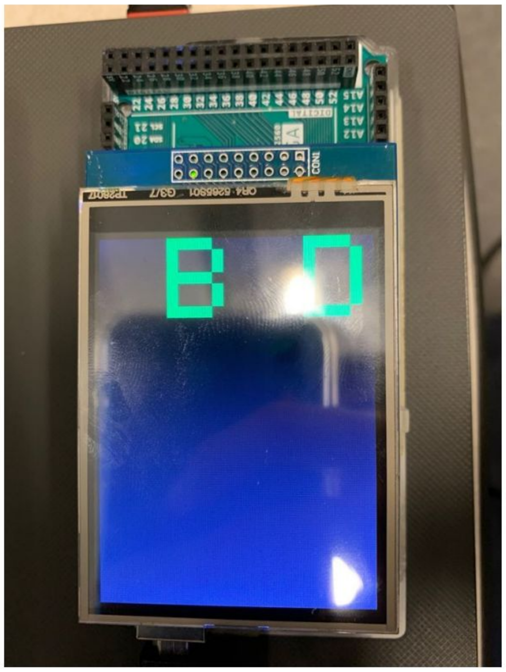
```
yes@yes-XPS:~/Documents/EE474_Introduction_to_embedded_system/Hw1$ ./two
Please enter a single digit between 0-9
7
The data is: 7
The data is: 7
```

* Solution: Since the pointer has already point to an integer's address, we can just use that address to store the input, meaning that in the function getData() we comment out the line 33 and 36. The input number will be directory stored in the address of myValue.

Serial communication (Third Part)
We need to first set up the same baud rate for each device. In our case, we use 9600 as our baud rate. And then in the hardware side, we need to link the tx on one device to the rx on the other device. And vice versa. When testing, we can use the serial panel on the Arduino IDE to check whether the device receive some message or not.

Part 2.





ERROR ANALYSIS:

Since the delay function is implemented using for loop, it's hard for us to make the delay time as we expected. Also, in order to make the delay time noticeable, we need to give a very big iteration number for the for loop. This will not be a good solution if we are going to implement in our future project, that required some accurate time delay function. A better solution will be using the delay() function in Arduino.

SUMMARY:
In the first part, we test out different types of displaying methods and incorporate with a delay function using for loop. This is not a general way to delay time but is enough for our task. Different kinds of displaying methods can result in different user experience. This gave us some insight about what will be the result if we use each of the method to implement.

In the debugging part, there are several useful tool such as gdb for us to detect the usage of computer resources. However, in our case, the code is too trivial and can easily find out the mistake in the code. Maybe in the future, we will still faced with the situation that we need to use gdb to debug.

In the serial part, we know that by giving different baud rate we can get different communication speed. However, this might still be constrain to the hardware limitation. But in our case, using 9600 is already enough because we are not asking for some quick responses.

CONCLUSION:

In this project, we learned more about the C programming, not only the architecture, but also some frequently made mistakes. On the other hand, we get used to Arduino and TFT. The functions we learned in the project, such as communication, displaying, and debugging will for sure be important for our future project. Even though all the tasks in this project are not too hard, but we still laid a solid foundation.

CONTRIBUTION:

Alex:
- Port code developed on CodeBlock to ATMega board and display with TFT.
- Establish serial communication between ATMega board and Uno board.
- List the spec of project and test cases.

Shawn:
- Develop C (Application 1 to 8) programs on CodeBlock.
- Explain the software implementation.
- Explain the error analysis.

Shih-Hao:
- Debug the given code.
- Elaborate abstract, introduction, analysis, summary and conclusion in report.

APPENDIX:

project1a-2019.c

```
// preprocessor directive to support printing to the display

#include <stdio.h>
#include <windows.h>
#include "Application8.h"
// the main program


int application1(void)
{
 int i=0;
 while (i<10){
     char A[5]={'A','B','C','D','\0'};
     i+=1;
     printf(A);
     Sleep(500);
     system("cls");
     Sleep(500);

 };
 return 0;
}

int application2(void)
{
 int i=0;
 while (i<10){
     char A[5]={'A','B','C','D','\0'};
     char B[5]={' ','B','C','D','\0'};
     char C[5]={' ',' ','C','D','\0'};
     char D[5]={' ',' ',' ','D','\0'};
     i+=1;
     printf(A);
     Sleep(300);
     system("cls");
     printf(B);
     Sleep(300);
     system("cls");
     printf(C);
     Sleep(300);
     system("cls");
     printf(D);
     Sleep(300);
     system("cls");
     Sleep(300);

 };
 return 0;
}

int application3(void)
{
 int i=0;
 int j=0;
 int k=0;
 char A[5]={'A',' ','C','\t','\0'};
 char B[5]={' ','B',' ','D','\0'};
 for (i=0;i<20;i++){

     if (i%2==0){
        printf(A);
        for(j=0;j<100000000;j++){};
     };

     if (i%2==1){
        printf(B);
        for(j=0;j<200000000;j++){};
     };


     system("cls");


 };
 return 0;
}

int application4(int delay_time_1,int delay_time_2)
{
 int i=0;
 int j=0;
 int k=0;
 char A[5]={'A',' ','C','\0'};
 char B[5]={' ','B',' ','D','\0'};
 for (i=0;i<20;i++){

     if (i%2==0){
        printf(A);
        for(j=0;j<delay_time_1*100000000;j++){};
     };

     if (i%2==1){
        printf(B);
        for(j=0;j<delay_time_2*100000000;j++){};
     };
     system("cls");


 };
 return 0;
}

void f1Data(unsigned long *delay1){
     int j=0;
     for(j=0;j<*delay1*100000000;j++){};}

void f2Clear(unsigned long *delay2){
    int j;
    for(j=0;j<*delay2*100000000;j++){};}



int application5(int delay_time_1,int delay_time_2)
{
 int i=0;

 char A[5]={'A',' ','C','\0'};
 char B[5]={' ','B',' ','D','\0'};


 for (i=0;i<20;i++){

     if (i%2==0){
        printf(A);
        f1Data(delay_time_1);

     };

     if (i%2==1){
        printf(B);
        f1Data(delay_time_1);
     };


     system("cls");
     f2Clear(delay_time_2);
 };
 return 0;
}

int application6(int *delay_time_1,int *delay_time_2)
{
 int i=0;

 char A[5]={'A',' ','C','\0'};
 char B[5]={' ','B',' ','D','\0'};


 for (i=0;i<20;i++){

     if (i%2==0){
        printf(A);
        f1Data(delay_time_1);

     };

     if (i%2==1){
        printf(B);
        f1Data(delay_time_1);
     };


     system("cls");
     f2Clear(delay_time_2);


 };
 return 0;
}

int application7(char A[],char B[],int *delay_time_1,int *delay_time_2)
{
 int i=0;

 for (i=0;i<20;i++){

     if (i%2==0){
        printf(A);
```

```
      f1Data(delay_time_1);

    };

    if (i%2==1){
      printf(B);
      f1Data(delay_time_1);
    };


    system("cls");
    f2Clear(delay_time_2);


 };
 return 0;
}

int main(void)
{
        //  declare, efine, and inbitializa some local variables
        int x =  9;
        int y = 10;
        int z =  0;
        float a = 0.0;

        //  preform a simple calculation
        z = x+y;

        //  print the results of the calculation to the display
        printf ("the sum of x and y is %d\n" ,z);

        //  ask the user for some data
        printf("please enter a value\n");

        //  get the data from the user
        //    the data will be a floating point number %f
        //    stored in the variable 'a'
        //    the & operator takes the address of the variable 'a'

        scanf ("%f", &a);

        //  remove the newline from input buffer
```

```
            //getchar();

            //  print the user data to the display
            //    the format will be xx.yy

            printf ("the data is %2.2f\n", a);
            //application1();
            int c=1;
            int d=2;
            char A[5]={'A',' ','C','\0'};
    char B[5]={' ','B',' ','D','\0'};
    application1();
    application2();
    application3();
    application4(c,d);
    application5(c,d);
    application6(&c,&d);
            application7(A,B,&c,&d);
            application8(&c,&d);
            return 0;
}
```

Application8.h

```
#ifndef _APPLICATION8_H_
#define _APPLICATION8_H_


#include <stdio.h>
#include <windows.h>


int application8(int *delay_time_1,int *delay_time_2)
{
 int i=0;

 char A[5]={'A',' ','C','\0'};
 char B[5]={' ','B',' ','D','\0'};


 for (i=0;i<20;i++){

     if (i%2==0){
        printf(A);
        f1Data(delay_time_1);

      };

      if (i%2==1){
        printf(B);
        f1Data(delay_time_1);
      };


      system("cls");
      f2Clear(delay_time_2);


  };
  return 0;
}

#endif
```

project1b-2019.c

```
#include <stdio.h>

//  this is a simple routine that demonstrates how to fill and diaplay an array of characters

void main(void)
{
        int i = 0;                                                      //  declare a
working variable

        char myArray[6];                                                //  declare a character
array

        for (i = 0; i <= 5; i++)                            //  fill array with characters
        {
                //  fill with the ascii characters A..F
                //  65 is the ascii value for A

                myArray[i]= 65+i;
        }

        for (i = 0; i <= 5; i++)                            //  display the array
        {
                printf("%c \n", myArray[i]);
        }

        printf("\n");

        return;
}
```

project1c-2019.c

```
#include <stdio.h>

//  function prototypes

//  get data from the user
void getData(int* aValuePtr);

void main (void)
{
        //  declare a shared variable and a pointer to it
        int myValue;
        int* myPtr = &myValue;                  //  let myPtr point to myValue

        //  get data from the user
        getData(myPtr);

        //  display the data as a character
        printf("The data is: %c \n", *myPtr);

}

//  prompt the user for some data and return it through a shared
//  variable pointed to by valuePtr
//
//  inputs:         pointer to a container in which to place the data
//  outputs:        none
//  function:       the routine accepts a pointer to a container in which to store data from a user,
//                                  it prompts for the data, accepts the data, displays it, and returns

void getData(int* valuePtr)
{
        //  declare a temp place to store the data
        // int tempValue;
        //  let valuePtr point to it
        // valuePtr = &tempValue;

        //  prompt for data
        printf("Please enter a single digit between 0-9 \n");

        //  get the data
        *valuePtr = getchar();

        //  display its value as a character
        printf("The data is: %c \n", *valuePtr);

        return;

}
```

project1d-2019.ino

```
// IMPORTANT: ELEGOO_TFTLCD LIBRARY MUST BE SPECIFICALLY
// CONFIGURED FOR EITHER THE TFT SHIELD OR THE BREAKOUT BOARD.
// SEE RELEVANT COMMENTS IN Elegoo_TFTLCD.h FOR SETUP.
//Technical support:goodtft@163.com

#include <Elegoo_GFX.h>    // Core graphics library
#include <Elegoo_TFTLCD.h> // Hardware-specific library

// The control pins for the LCD can be assigned to any digital or
// analog pins...but we'll use the analog pins as this allows us to
// double up the pins with the touch screen (see the TFT paint example).
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0

#define LCD_RESET A4 // Can alternately just connect to Arduino's reset pin

// When using the BREAKOUT BOARD only, use these 8 data lines to the LCD:
// For the Arduino Uno, Duemilanove, Diecimila, etc.:
//   D0 connects to digital pin 8  (Notice these are
//   D1 connects to digital pin 9   NOT in order!)
//   D2 connects to digital pin 2
//   D3 connects to digital pin 3
//   D4 connects to digital pin 4
//   D5 connects to digital pin 5
//   D6 connects to digital pin 6
//   D7 connects to digital pin 7
// For the Arduino Mega, use digital pins 22 through 29
// (on the 2-row header at the end of the board).

// Assign human-readable names to some common 16-bit color values:
#define          BLACK   0x0000
#define          BLUE    0x001F
#define          RED     0xF800
#define          GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

Elegoo_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
// If using the shield, all control and data lines are fixed, and
// a simpler declaration can optionally be used:
// Elegoo_TFTLCD tft;



void setup(void) {
  Serial.begin(9600);
  Serial.println(F("TFT LCD test"));


#ifdef USE_Elegoo_SHIELD_PINOUT
  Serial.println(F("Using Elegoo 2.4\" TFT Arduino Shield Pinout"));
#else
  Serial.println(F("Using Elegoo 2.4\" TFT Breakout Board Pinout"));
#endif

  Serial.print("TFT size is "); Serial.print(tft.width()); Serial.print("x"); Serial.println(tft.height());

  tft.reset();

   uint16_t identifier = tft.readID();
   if(identifier == 0x9325) {
    Serial.println(F("Found ILI9325 LCD driver"));
  } else if(identifier == 0x9328) {
    Serial.println(F("Found ILI9328 LCD driver"));
  } else if(identifier == 0x4535) {
    Serial.println(F("Found LGDP4535 LCD driver"));
  }else if(identifier == 0x7575) {
    Serial.println(F("Found HX8347G LCD driver"));
  } else if(identifier == 0x9341) {
    Serial.println(F("Found ILI9341 LCD driver"));
  } else if(identifier == 0x8357) {
    Serial.println(F("Found HX8357D LCD driver"));
  } else if(identifier==0x0101)
  {
      identifier=0x9341;
       Serial.println(F("Found 0x9341 LCD driver"));
  }
  else if(identifier==0x1111)
  {
      identifier=0x9328;
       Serial.println(F("Found 0x9328 LCD driver"));
  }
  else {
    Serial.print(F("Unknown LCD driver chip: "));
    Serial.println(identifier, HEX);
    Serial.println(F("If using the Elegoo 2.8\" TFT Arduino shield, the line:"));
    Serial.println(F("  #define USE_Elegoo_SHIELD_PINOUT"));
    Serial.println(F("should appear in the library header (Elegoo_TFT.h)."));
    Serial.println(F("If using the breakout board, it should NOT be #defined!"));
    Serial.println(F("Also if using the breakout, double-check that all wiring"));
    Serial.println(F("matches the tutorial."));
    identifier=0x9328;

  }
  tft.begin(identifier);

}

  int i=0;  //integer for looping
  char myArray[10]; //array to store numbers going to be displayed

void loop(void)
{
  tft.fillScreen(BLACK); //set the screen color to black
  unsigned long start = micros(); //record the start time
  tft.setCursor(0, 0); //set cursor position to the origin

  tft.setTextColor(GREEN); tft.setTextSize(2); //set text color to green, size to 2
  for (i = 0; i <=9; i++)
  {
    myArray[i]= 48+i; //fill ascii code of 0 to 9 in myArray
  }

  for (i = 9; i >= 0; i--)
  {
    tft.print(myArray[i]); //show values in myArray
    delay(1000); //delay for 1 second
  }

  delay(1000);delay(1000);delay(1000); //delay for 3 seconds
}
```

project1d_7-2019.ino

```
// IMPORTANT: ELEGOO_TFTLCD LIBRARY MUST BE SPECIFICALLY
// CONFIGURED FOR EITHER THE TFT SHIELD OR THE BREAKOUT BOARD.
// SEE RELEVANT COMMENTS IN Elegoo_TFTLCD.h FOR SETUP.
//Technical support:goodtft@163.com

#include <Elegoo_GFX.h>    // Core graphics library
#include <Elegoo_TFTLCD.h> // Hardware-specific library

// The control pins for the LCD can be assigned to any digital or
// analog pins...but we'll use the analog pins as this allows us to
// double up the pins with the touch screen (see the TFT paint example).
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0

#define LCD_RESET A4 // Can alternately just connect to Arduino's reset pin

// When using the BREAKOUT BOARD only, use these 8 data lines to the LCD:
// For the Arduino Uno, Duemilanove, Diecimila, etc.:
//   D0 connects to digital pin 8  (Notice these are
//   D1 connects to digital pin 9   NOT in order!)
//   D2 connects to digital pin 2
//   D3 connects to digital pin 3
//   D4 connects to digital pin 4
//   D5 connects to digital pin 5
//   D6 connects to digital pin 6
//   D7 connects to digital pin 7
// For the Arduino Mega, use digital pins 22 through 29
// (on the 2-row header at the end of the board).

// Assign human-readable names to some common 16-bit color values:
#define         BLACK   0x0000
#define         BLUE    0x001F
#define         RED     0xF800
#define         GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

Elegoo_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
// If using the shield, all control and data lines are fixed, and
// a simpler declaration can optionally be used:
// Elegoo_TFTLCD tft;



void setup(void) {
  Serial.begin(9600);
  Serial.println(F("TFT LCD test"));


#ifdef USE_Elegoo_SHIELD_PINOUT
  Serial.println(F("Using Elegoo 2.4\" TFT Arduino Shield Pinout"));
#else
  Serial.println(F("Using Elegoo 2.4\" TFT Breakout Board Pinout"));
#endif

  Serial.print("TFT size is "); Serial.print(tft.width()); Serial.print("x"); Serial.println(tft.height());

  tft.reset();

   uint16_t identifier = tft.readID();
   if(identifier == 0x9325) {
    Serial.println(F("Found ILI9325 LCD driver"));
  } else if(identifier == 0x9328) {
    Serial.println(F("Found ILI9328 LCD driver"));
  } else if(identifier == 0x4535) {
    Serial.println(F("Found LGDP4535 LCD driver"));
  }else if(identifier == 0x7575) {
    Serial.println(F("Found HX8347G LCD driver"));
  } else if(identifier == 0x9341) {
    Serial.println(F("Found ILI9341 LCD driver"));
  } else if(identifier == 0x8357) {
    Serial.println(F("Found HX8357D LCD driver"));
  } else if(identifier==0x0101)
  {
      identifier=0x9341;
       Serial.println(F("Found 0x9341 LCD driver"));
  }
  else if(identifier==0x1111)
  {
      identifier=0x9328;
       Serial.println(F("Found 0x9328 LCD driver"));
  }
  else {
    Serial.print(F("Unknown LCD driver chip: "));
    Serial.println(identifier, HEX);
    Serial.println(F("If using the Elegoo 2.8\" TFT Arduino shield, the line:"));
    Serial.println(F("  #define USE_Elegoo_SHIELD_PINOUT"));
    Serial.println(F("should appear in the library header (Elegoo_TFT.h)."));
    Serial.println(F("If using the breakout board, it should NOT be #defined!"));
    Serial.println(F("Also if using the breakout, double-check that all wiring"));
    Serial.println(F("matches the tutorial."));
    identifier=0x9328;

  }
  tft.begin(identifier);

  tft.fillScreen(BLACK); //set the screen color to black
  unsigned long start = micros(); //record the start time
  tft.setCursor(0, 0); //set cursor position to the origin

  tft.setTextColor(GREEN); tft.setTextSize(10); //set text color to green, size to 10
}

  int i=0;  //integer for looping
  long unsigned int c=1000, d=2000;  //c is delay time for fiData, d is delay time for f2clear
  char A[5]={'A',' ','C','\0'};      //The String 1 shown on the display
  char B[5]={' ','B',' ','D','\0'};  //The String 2 shown on the display

void f1Data(unsigned long *delay1){
     int j=0;
     for(j=0;j<*delay1*100000000;j++){};}  // Delay time for displaying

void f2Clear(unsigned long *delay2){
    int j;
    for(j=0;j<*delay2*100000000;j++){};}  //Delay time for diappearing

void loop(void)
{
  for (i=0;i<20;i++){

     if (i%2==0){
        tft.print(A); //Print
        f1Data(&c);                      //Call function f1Data to show a string A in a duration of delay 1

     };

     if (i%2==1){
        tft.print(B);
        f1Data(&c);                     //Call function f1Data to show a string B in a duration of delay 1

     };
     //system("cls");
     tft.fillScreen(BLACK);
     tft.setCursor(0, 0);
     f2Clear(&d);                     //Call function f1clear to popstone the duration time of
disappearance

  }
}
```