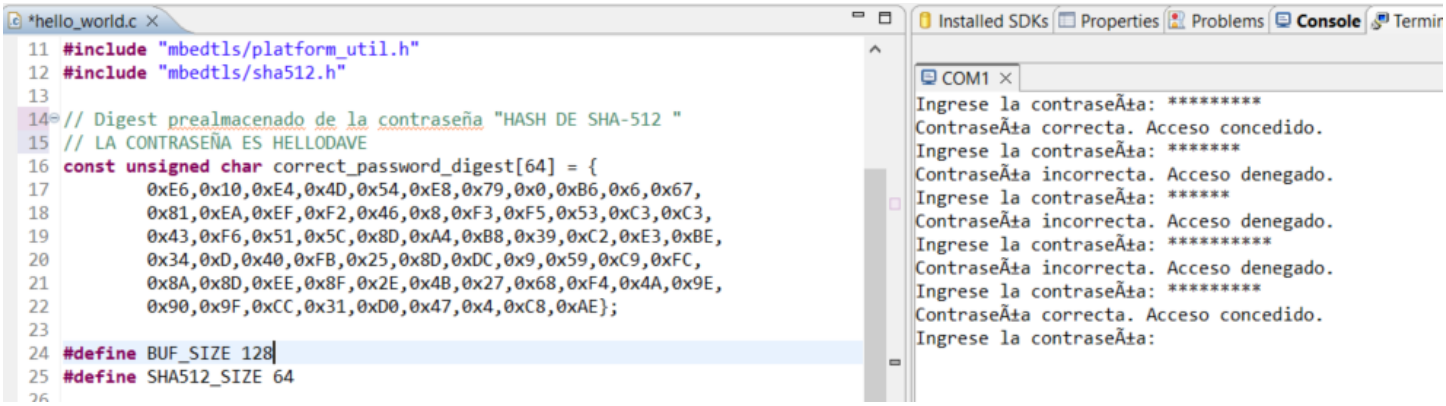
EL ARCUIVO “helloword” es el main en cada proyecto

EJECUCION DE implementación de los algoritmos de SHA512 tarjeta OKDo.

```
/*****************************************************************************************************
*****************************************************************************/
/*****************          PASS WORD PARA UN HASH DE SHA 512
*********************************************/
#include "fsl_device_registers.h"
#include "fsl_debug_console.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"

#include "fsl_power.h"
#include "mbedtls/platform.h"
#include "mbedtls/platform_util.h"
#include "mbedtls/sha512.h"

// Digest prealmacenado de la contraseña "HASH DE SHA-512 "
// LA CONTRASEÑA ES HELLODAVE
const unsigned char correct_password_digest[64] = {
        0xE6,0x10,0xE4,0x4D,0x54,0xE8,0x79,0x0,0xB6,0x6,0x67,
        0x81,0xEA,0xEF,0xF2,0x46,0x8,0xF3,0xF5,0x53,0xC3,0xC3,
        0x43,0xF6,0x51,0x5C,0x8D,0xA4,0xB8,0x39,0xC2,0xE3,0xBE,
        0x34,0xD,0x40,0xFB,0x25,0x8D,0xDC,0x9,0x59,0xC9,0xFC,
        0x8A,0x8D,0xEE,0x8F,0x2E,0x4B,0x27,0x68,0xF4,0x4A,0x9E,
        0x90,0x9F,0xCC,0x31,0xD0,0x47,0x4,0xC8,0xAE};

#define BUF_SIZE 128
#define SHA512_SIZE 64

unsigned char sha512sum[SHA512_SIZE];
// Inicialización del contexto de SHA-512
mbedtls_sha512_context sha512_ctx;

void resetPasswordBuffer(char *buffer, size_t size)
{
    for (size_t i = 0; i < size; ++i)
    {
        buffer[i] = '\0';
    }
}

void calculateSHA512(const char *input, size_t size, unsigned char *output)
{
    mbedtls_sha512_starts_ret(&sha512_ctx, 0);
    mbedtls_sha512_update_ret(&sha512_ctx, (const unsigned char *)input, size);
    mbedtls_sha512_finish_ret(&sha512_ctx, output);
}

int compareDigests(const unsigned char *digest1, const unsigned char *digest2, size_t size)
{
    for (size_t i = 0; i < size; ++i)
    {
        if (digest1[i] != digest2[i])
        {
            return 0; // No son iguales
        }
    }
    return 1; // Son iguales
}

int main(void)
{
    char ch;
```

```
    char passwordBuffer[BUF_SIZE];
    unsigned char userPasswordDigest[SHA512_SIZE];

    /* Inicialización del hardware y configuraciones iniciales */
    POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();
    CRYPTO_InitHardware();

    // Inicialización del contexto SHA-512
    mbedtls_sha512_init(&sha512_ctx);

    while (1)
    {
        resetPasswordBuffer(passwordBuffer, BUF_SIZE);

        PRINTF("Ingrese la contraseña: ");
        uint8_t index = 0;
        while (1)
        {
            ch = GETCHAR();
            if (ch == '\r' || index >= BUF_SIZE - 1)
            {
                break;
            }
            PUTCHAR('*'); // Muestra "*" en lugar del caracter real para mayor seguridad
            passwordBuffer[index++] = ch;
        }

        // Calcula el hash SHA-512 de la contraseña ingresada
        calculateSHA512(passwordBuffer, index, userPasswordDigest);

        // Compara el hash de la contraseña ingresada con el digest prealmacenado
      if (compareDigests(userPasswordDigest, correct_password_digest, SHA512_SIZE))
        {
            PRINTF("\n\rContraseña correcta. Acceso concedido.\r\n");
        }
        else
        {
            PRINTF("\n\rContraseña incorrecta. Acceso denegado.\r\n");
        }
    }
}
```

RESULTADO DE SHA 512 VERIFICANDO HASH



EJECUCION DE implementación de los algoritmos de SHA256 en la tarjeta OKDo.

```
/*****************************************************************************************************
*****************************************************************************/
/******************          PASS WORD PARA UN HASH SHA 256
*************************************************/
#include "fsl_device_registers.h"
#include "fsl_debug_console.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "board.h"

#include "fsl_power.h"
#include "mbedtls/platform.h"
#include "mbedtls/platform_util.h"
#include "mbedtls/sha256.h"

// Digest prealmacenado de la contraseña "SHA 256 HASH" LA CONTRASEÑA ES HELLODAVE
const unsigned char correct_password_digest[32] = {
        0xB9,0xBA,0xF6,0x3C,0x97,0xB6,0x86,0x4C,0x8,
        0xAE,0x50,0xD5,0xE6,0xEE,0xD6,0x62,0x66,0x8E,
        0x19,0xBB,0x9C,0x19,0x8F,0xDF,0x21,0x7F,0x26,
        0x9A,0x91,0xC3,0x24,0x83
};

#define BUF_SIZE 128
#define SHA256_SIZE 32

unsigned char sha256sum[SHA256_SIZE];

// Inicialización del contexto de SHA-256
mbedtls_sha256_context sha256_ctx;

void resetPasswordBuffer(char *buffer, size_t size)
{
    for (size_t i = 0; i < size; ++i)
    {
        buffer[i] = '\0';
    }
}

void calculateSHA256(const char *input, size_t size, unsigned char *output)
{
    mbedtls_sha256_starts_ret(&sha256_ctx, 0);
    mbedtls_sha256_update_ret(&sha256_ctx, (const unsigned char *)input, size);
    mbedtls_sha256_finish_ret(&sha256_ctx, output);
}

int compareDigests(const unsigned char *digest1, const unsigned char *digest2, size_t size)
{
    for (size_t i = 0; i < size; ++i)
    {
        if (digest1[i] != digest2[i])
        {
            return 0; // No son iguales
        }
    }
    return 1; // Son iguales
}

int main(void)
{
```

```
    char ch;
    char passwordBuffer[BUF_SIZE];
    unsigned char userPasswordDigest[SHA256_SIZE];

    /* Inicialización del hardware y configuraciones iniciales */
    POWER_SetBodVbatLevel(kPOWER_BodVbatLevel1650mv, kPOWER_BodHystLevel50mv, false);
    CLOCK_AttachClk(BOARD_DEBUG_UART_CLK_ATTACH);
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitDebugConsole();
    CRYPTO_InitHardware();

    // Inicialización del contexto SHA-256
    mbedtls_sha256_init(&sha256_ctx);

    while (1)
    {
        resetPasswordBuffer(passwordBuffer, BUF_SIZE);

        PRINTF("Ingrese la contraseña: ");
        uint8_t index = 0;
        while (1)
        {
            ch = GETCHAR();
            if (ch == '\r' || index >= BUF_SIZE - 1)
            {
                break;
            }
            PUTCHAR('*'); // Muestra "*" en lugar del caracter real para mayor seguridad
            passwordBuffer[index++] = ch;
        }

        // Calcula el hash SHA-256 de la contraseña ingresada
        calculateSHA256(passwordBuffer, index, userPasswordDigest);

        // Compara el hash de la contraseña ingresada con el digest prealmacenado
        if (compareDigests(userPasswordDigest, correct_password_digest, SHA256_SIZE))
        {
            PRINTF("\n\rContraseña correcta. Acceso concedido.\r\n");
        }
        else
        {
            PRINTF("\n\rContraseña incorrecta. Acceso denegado.\r\n");
        }
    }
}
```

RESULTADO DE ESA EJECUCION