ELEC-C7420 Basic Principles in Networking Spring 2022

Assignment IV: Implementing Hash functions for Digital Signatures

PSALTAKIS GEORGIOS

Aalto University
School of Engineering

Goals of the experiment

The goals of the experiment are to understand how the MD5 and SHA-1 hash functions work. We also implement learn how these two basic and very important hash functions work by implementing them on arduino. The basics is to understand the differences between those two functions as well as how they actually encode the messages.

Experimental Setup

The setup of the experiment is simple. We have implemented the code and we write a simple text in the message of the code to encode it either using MD5 or either using SHA1.

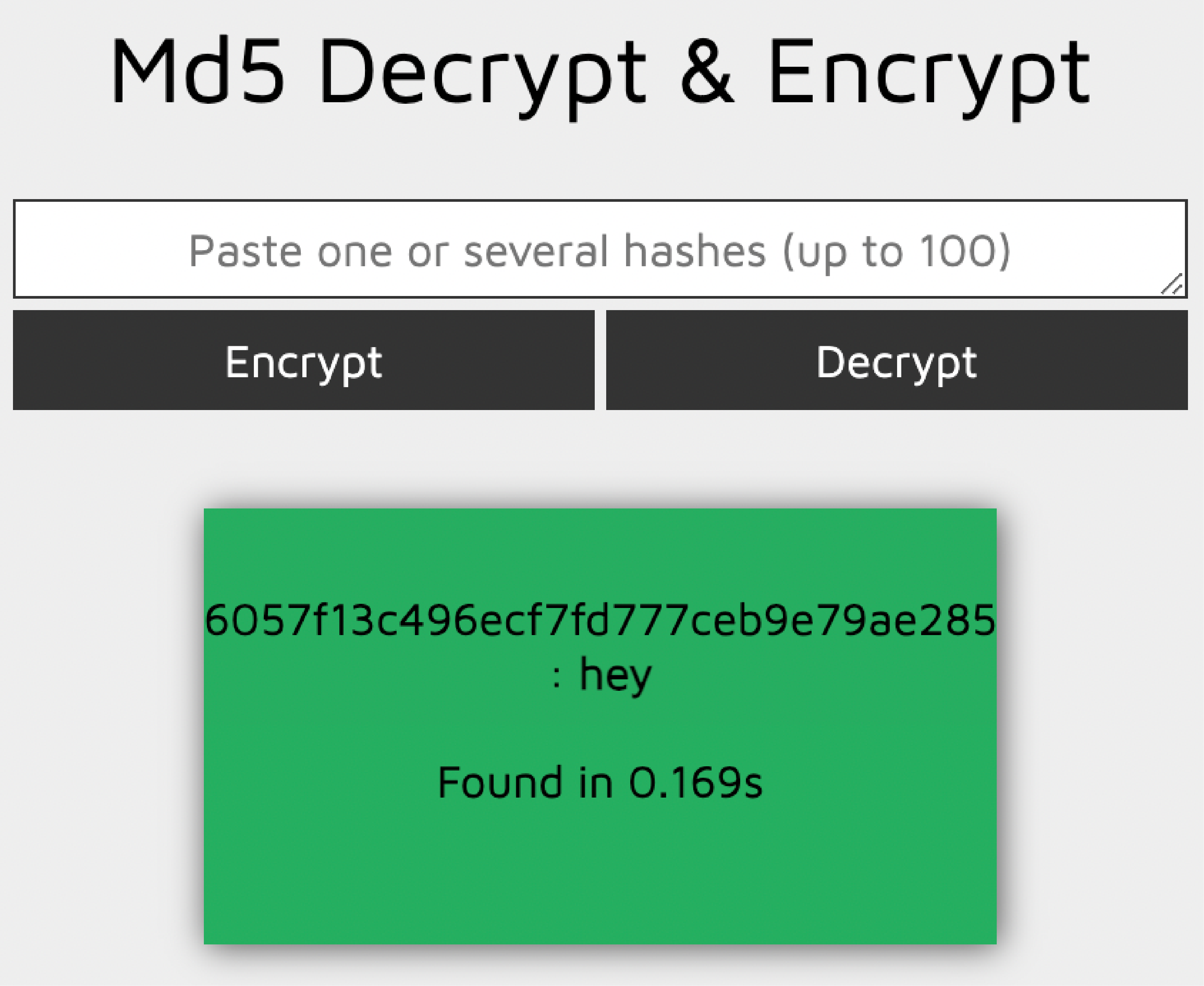
Since we use SHA1 and MD5 we need to add the corresponding libraries to arduino for them to work. These are of course the <MD5.h> for the md5 encoding the <Hash.h> for the basics before encoding on sha1 as well as the <SHA1.h> there are allot of different variations for there libraries but at the end of the day the aim for the same result with couple different technicalities. We also have included the <ArduinoBearSSL.h> because its needed for the SHA1 encoding.

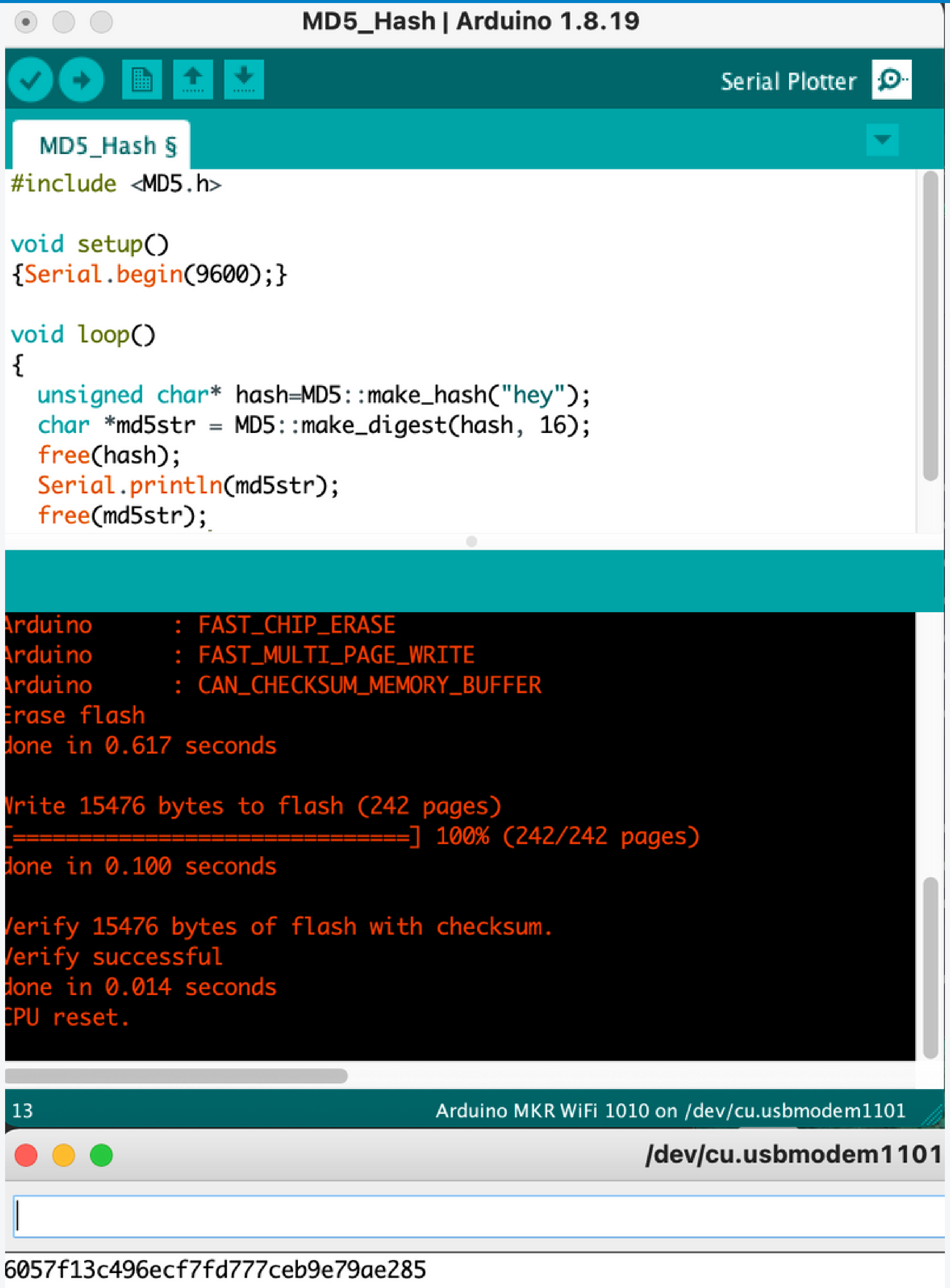
This code has been uploaded to the Arduino without problems with both of the cases having the message "hey" hardcoded . This of course gives us the correct results

SHA1 : 7f550a9f4c44173a37664d938f1355f0f92a47a7
MD5 : 6057f13c496ecf7fd777ceb9e79ae285

Results & Conclusion & Annex

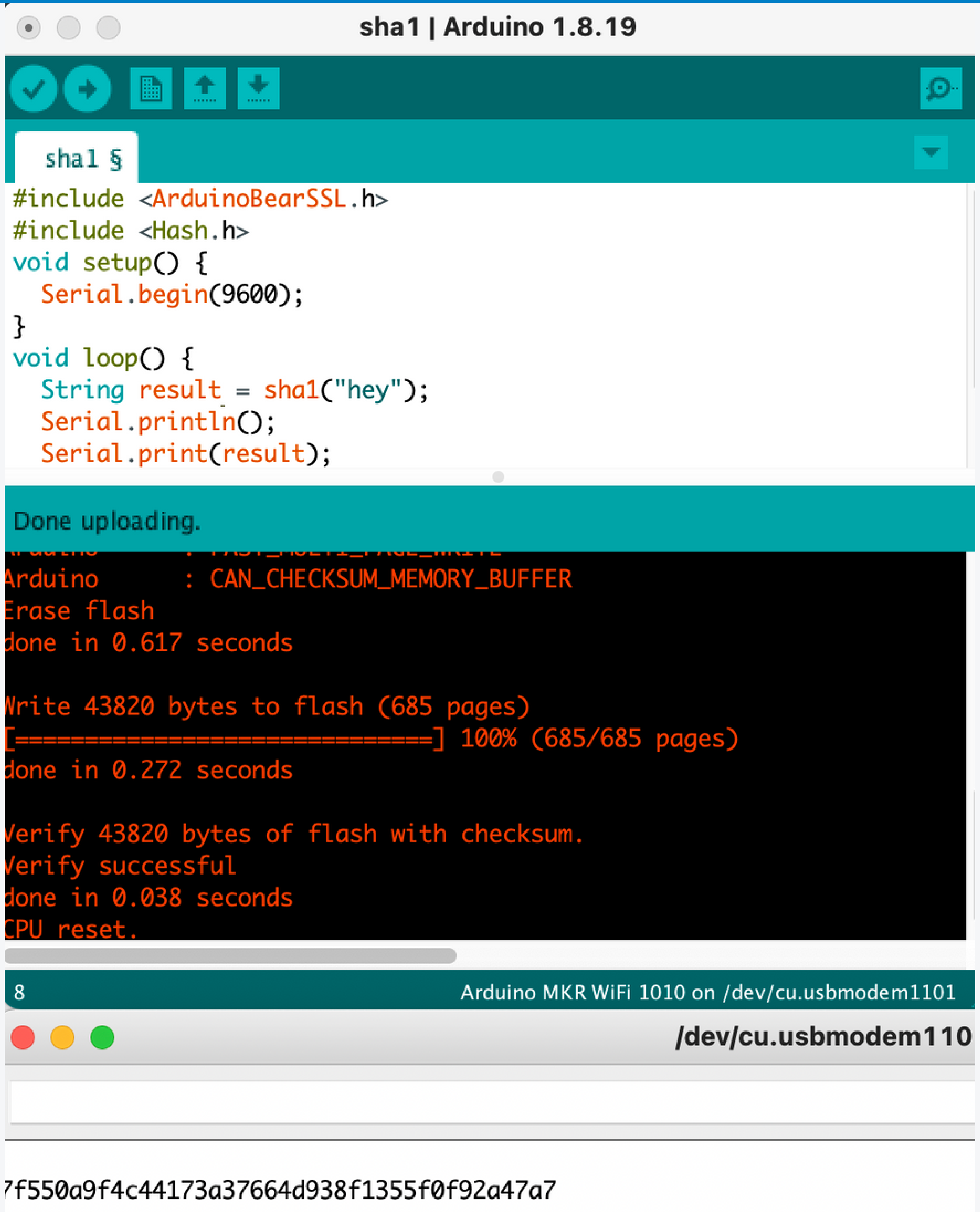
Creating successful compilation of sketch and upload without an error and compilation of the actual results needed from the serial monitor




MD5_Hash | Arduino 1.8.19
Serial Plotter
MD5_Hash §
#include <MD5.h>

void setup()
{Serial.begin(9600);}

void loop()
{
  unsigned char* hash=MD5::make_hash("hey");
  char *md5str = MD5::make_digest(hash, 16);
  free(hash);
  Serial.println(md5str);
  free(md5str);
Arduino : FAST_CHIP_ERASE
Arduino : FAST_MULTI_PAGE_WRITE
Arduino : CAN_CHECKSUM_MEMORY_BUFFER
Erase flash
done in 0.617 seconds
Write 15476 bytes to flash (242 pages)
[==============================] 100% (242/242 pages)
done in 0.100 seconds
Verify 15476 bytes of flash with checksum.
Verify successful
done in 0.014 seconds
CPU reset.
13
Arduino MKR WiFi 1010 on /dev/cu.usbmodem1101
/dev/cu.usbmodem1101
6057f13c496ecf7fd777ceb9e79ae285


sha1 | Arduino 1.8.19
sha1 §
#include <ArduinoBearSSL.h>
#include <Hash.h>
void setup() {
  Serial.begin(9600);
}
void loop() {
  String result = sha1("hey");
  Serial.println();
  Serial.print(result);
Done uploading.
Arduino : CAN_CHECKSUM_MEMORY_BUFFER
Erase flash
done in 0.617 seconds
Write 43820 bytes to flash (685 pages)
[==============================] 100% (685/685 pages)
done in 0.272 seconds
Verify 43820 bytes of flash with checksum.
Verify successful
done in 0.038 seconds
CPU reset.
8
Arduino MKR WiFi 1010 on /dev/cu.usbmodem1101
/dev/cu.usbmodem110
7f550a9f4c44173a37664d938f1355f0f92a47a7

Answer of the given questions

- Of the two mentioned hash function, would you use one for Security Application? Why? If not, provide an alternative.

Both of these hash functions have been secure in the past. But thats not the case anymore for both of them. The MD5 is no longer secure since 2011 by an article published "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms," which mentioned allot of recent attacks on md5 hashes making them no longer secure for anything since they can be cracked in seconds in these times.The SHA-1 has not been secure enough since 2005 and since 2010 it has been recommended across the board to not be used for anything secure. The replacement for both of these hash function is easily the SHA-256 which according to recent research is considerably more secure than its predaccesors md5 and sha1.

- Please explain in brief what makes hash functions resistant to attacks. Provide an exemplary brief case study.

The main thing that makes the hash functions resistant to attacks is their collision resistance of course. By collision resistance we mean the property of cryptographic hash functions that is hard to find two inputs that hash to the same output. that means two inputs such as a and b that are y(a) = y(b). The harder this is in a hash function the more collision resistant that hash function is. Of course any hash function with more inputs than outputs will have collisions. For example the MD5 was secure enough but researchers were able to generate two different files with the same MD5 hash value ! This in fact is a collision attack and it makes the MD5 less secure

- Provide a comparison between MD5 and SHA-1. Overall, which one do you think performs better than the other one?

MD5	SHA1
MD5 can have messages upt to 128 bits	while sha1 can have 160 bits message digest
to make the initial messages the aggresor needs 2^128 operations	where the sha1 needs 2^160 making it more difficult to seek out
MD5 is simpler than sha1	SHA1 is more complex
MD5 provides poor security	where SHA1 provides a bit better security but still not good enough.
if you try to seek two messages with identical digest it would need to perform 2^64 operations	Where Sha-1 would need 2^80 operations

Both of them are bad at this time since they have been cracked. and since md5 is 7.6% slower than sha1 and less secure i woukld prefer the sha1 even though its more complex and requires more operations.

- What does it mean for a hash algorithm to be broken?

the main thing that if it happens to any hasking function it would be completely broken is if (x == y && hash(x) != hash(y)) this basically means that we know a y(s1) and we find a second s2 that is equal to y(s1)=y(s2) or that we are able to find s1 and s2 where y(s1)=y(s2) with s1 differnt than s2. Once the collisions are found for the specific hash algorithm that satifies the above the algorithm is no longer secure for cryptographic use . When this happens the hash algorithm is considered broked or dead like the MD5 since its no longer secure .

CODE FOR THE LIBRARIES INSTALLED.

The md5 is heavily based on that one. while the movents are the same for every md5 available in the internet.

https://github.com/tzikis/ArduinoMD5/blob/master/MD5.cpp

```
typedef struct {
        MD5_u32plus lo, hi;
        MD5_u32plus a, b, c, d;
        unsigned char buffer[64];
        MD5_u32plus block[16];
} MD5_CTX;

class MD5
{
public:
        MD5();
        static unsigned char* make_hash(char *arg);
        static unsigned char* make_hash(char *arg,size_t size);
        static char* make_digest(const unsigned char *digest, int len);
        static const void *body(void *ctxBuf, const void *data, size_t size);
        static void MD5Init(void *ctxBuf);
```

While we have the movements on the cpp code file the actual calls are happening at the h file as we can see here on the side. these are the calls to the cpp file which are then called on the ino file. The code for the arduino is above in printscreen.

For the SHA1 we just used the basic hash packet provided in the library as well as the sha1. I based the code on a tutorial for esp8266

http://www.esp8266learning.com/a-look-at-sha-1-and-esp8266.php

https://github.com/gcc-mirror/gcc/blob/master/include/sha1.h

```
class Sha1Wrapper : public Print
{
    public:
        void init(void);
        uint8_t * result(void);
#ifdef SHA1_ENABLE_HMAC
        void initHmac(const uint8_t * secret, unsigned int sec
        uint8_t * resultHmac(void);
#endif
        virtual size_t write(uint8_t);
        using Print::write;
    private:
        struct sha1_hasher_s _hasher;

};
```

Same thing here we have the movements on the cpp code file the actual calls are happening at the h file as we can see here on the side. these are the calls to the cpp file which are then called on the ino file . I am not including the whole code of the libraries cause it would take up at least 10 pages. But the idea is the same. the movements are done on the cpp files and called on the h libraries and then we can implement them on our ino arduino