# Assignment 1 – Cryptography Report

**Section 1: Goals of the experiment**

The main goal of this experiment is to successfully implement a simple Vigenère cipher with an Arduino board. Some sub goals are to learn how to write an algorithm and become familiar with the Arduino programming environment.

**Section 2: Experimental setup**

The hardware setup consisted of an Arduino MKR WiFi 1010 connected via USB type A to type B to a laptop. We utilized Arduino's IDE to program and upload the code to the board.

For the software, we used an array to store the key and read input from the user using the serial monitor. The key includes the alphabet positions of letters of "SNAKE". When there is input from the serial monitor, the program will add one of the elements from key to the byte. If the byte is greater than 90 (decimal value of 'Z'), the program will subtract 26 from the byte value since there are 26 letters in the English alphabet. Table 1 shows this process which is encryption of the message.

To test the accuracy of the algorithm, we hand-calculated the message in Table 1 to compare our results with and also focused on the encryption process first.

Key = (18, 13, 0, 10, 4) <- alphabet letter positions of the word 'SNAKE'

| Input letter (decimal) | 77 (M) | 69 (E) | 69 (E) | T (84) | | M(77) | E (69) |
|---|---|---|---|---|---|---|---|
| Key value | +18 | +13 | +0 | +10 | | +4 | +18 |
| Total byte value | 95 | 82 | 69 | 94 | | 81 | 87 |
| -26 if greater than 90 (Z) | -26 | | | -26 | | | |
| Final Total | 69 (E) | 82 (R) | 69 (E) | 68 (D) | | 81 (Q) | 87 (W) |

Table 1 Encryption Calculation of "MEET ME" to "ERED QW"

After receiving successful results from the encryption process, we decided to implement decryption next. Decryption is the exact opposite of the encryption process. Instead of adding the key values and subtracting 26, the process would be to subtract the key values and add 26.

| Input letter (decimal) | 69 (E) | 82 (R) | 69 (E) | 68 (D) | | 81 (Q) | 87 (W) |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Key value | -18 | -13 | -0 | -10 | | -4 | -18 |
| Total byte value | 52 | 69 | 69 | 58 | | 77 | 69 |
| +26 if less than 65 (A) | +26 | | | +26 | | | |
| Final Total | 77 (M) | 69 (E) | 69 (E) | 84 (T) | | M(77) | E (69) |

Table 2 Decryption Calculation of “ERED QW” back to “MEET ME”

To make the code more efficient, we utilized a “tag” to switch between encryption and decryption. In other words, this means that the user would have to input regular text (“MEET ME”) then the ciphered text (ERED QW) to get the correct outputs.

**Section 3: Results and Conclusion**

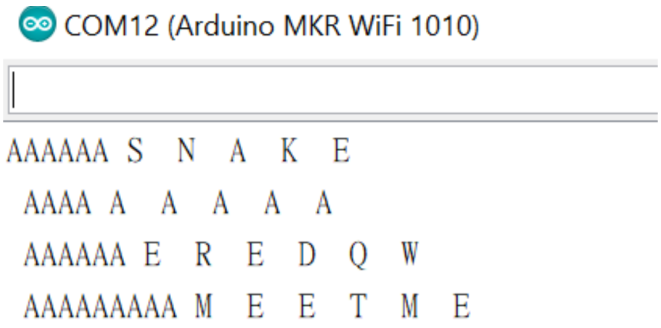
With the code that we wrote, provided in Appendix A, we were able to encrypt and decrypt user input from the serial monitor. The figure below shows the inputs, outputs and a screen capture of the serial monitor showing the usability of our code. The extra ‘A’s before each output is a subprogram called “establishContact” that was called to ensure that a connection between the computer and arduino board.

1.Input: AAAAA ---> Output: SNAKE

2.Input: SNAKE ---> Output: AAAAA

3.Input: MEETME ---> Output: EREDQW

4.Input: EREDQW ---> Output: MEETME




Since this was our first time programming in Arduino, one of challenges we faced was Serial monitor user input. Initially, after writing a simple draft of the program, we ran the program and received some unfamiliar and inconsistent outputs. To find the source of the problem, we decided to print the input and output together and found 2-3 extra characters. These characters are not within the ASCII values of the alphabet so to solve this issue we added the if statement to filter out unfamiliar characters (see “if (inByte >= 'A' && inByte <='Z')” in the Appendix).

In conclusion, we were able to successfully implement the Vigenre cipher to work with user input taken from the Serial monitor of an Arduino board.

**Section 4: Appendix**

```
char inByte = 0; // serial input and output character
int key[5] = {18, 13, 0, 10, 4};  //SNAKE
int ndx = 0;  //index for key
boolean pre_state = 0;
boolean state = 0;// Determine when to switch the functions "encryption" and "decryption".
int tag=-1;//The tag is for operate "encryption" and "decryption".

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600); // initialize the serial port at 9600 baud
  while (!Serial) {
  ; // wait for serial port to connect
  }
  establishContact(); // wait for incoming data
} /* setup */

void loop() {
  // put your main code here, to run repeatedly:
  // This is a simple Vigenère cipher algorithm
  // If you run the algorithm twice, you get back the original message
  // Example: ABC -> SOC -> ABC
  if (Serial.available() > 0) // if you have data input
  {
    state=1; //Change the state to 1 when we input a new text.
    if (state!=pre_state){ // If state!=pre_state, switch the function (state==1 & pre_state==0)
      tag=-tag;//You can see how the tag operated below
      ndx = 0;//Initial the ndx to 0 every time when switching the function
  }
  //if the state doesn't need to be changed,it operate the same function as the previous one

    inByte = Serial.read(); // read one byte of input
    if (inByte >= 'A' && inByte <='Z') //filter input
    {
      inByte += key[ndx]*tag; //when tag=1--->encryption--->inByte += key[ndx]
// when tag=-1--->decryption--->inByte -= key[ndx]
      if (inByte > 'Z' or inByte <'A')//modify the char when it's out of range
      {
        inByte -= 26*tag;/*when tag=1--->encryption--->inByte -= 26 / when tag=-1--->decryption---
>inByte += 26*/
      }

      if(ndx >= (5-1))//reuse the key for encryption and decryption
```

```
      {
        ndx = 0;
      }
      else
      {
        ndx++;
      }
    }//if inbyte is A-Z


    Serial.print(inByte); // write the encrypted character back
  } // if Serial.available() > 0
  establishContact();
} /* loop */

void establishContact()
{
  // write 'A' repeatedly until you receive data from the host
  Serial.print(" ");
  while (Serial.available() <= 0)
  {
    state = 0; //after input a text, reset the state until we input a new text
    Serial.print('A'); // write 'A' to the host
    delay(1000); // this delay is optional

  } // while Serial.available() <= 0
  Serial.print(" ");
  pre_state = state;//store the state to pre_state after every operation for each character.

} // establishContact()
```