**TEMASEK POLYTECHNIC  
SCHOOL OF INFORMATICS & IT**

**Diploma in Cyber Digital and Security**

**AY2018/2019**

Final Report

**Project Particulars**

|  |  |
| --- | --- |
| **Supervisor** | Dr Yap Chern Nam |
| **Project Number** |  |
| **Project Title** | Smart Lighting |

**Project Team’s Particulars**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student Number** | **Student Name** | **Email Address** | **Mobile No.** |
| **1602444J** | **Yeo Jie Kai** | **1602444J@student.tp.edu.sg** | **93728280** |

**COVER PAGE**

|  |
| --- |
| **DIPLOMA IN CYBER & DIGITAL SECURITY**    **Project Title**: Smart Lighting  **Submitted by**:  **Yeo Jie Kai 1602444J**    **Date:**  19/11/2018  **“By submitting this work, we are declaring that we are the originator(s) of this work and that all other original sources used in this work has been appropriately acknowledged.**  **We understand that plagiarism is the act of taking and using the whole or any part of another person’s work and presenting it as our own without proper acknowledgement.**  **We also understand that plagiarism is an academic offence and that disciplinary action will be taken for plagiarism.”**  **Name and Signature of student: Yeo Jie Kai** |

# **Acknowledgement**

I’m much obligated to my supervisor, Dr Yap Chern Nam, for the guidance he has given me for the span of my Major Project.

There were many obstacles and challenges I faced during the project and Dr Yap was always there to help me validate and provide feedback for my ideas and solutions, to ensure that I am on the correct approach to the project.

This project has given me the opportunity to demonstrate my skills and knowledge, by implementing security for hardware, and to grasp a sense of how security can be implemented in a real-world environment.

Table of Contents

[**Acknowledgement** 2](#_Toc126152)

[**Abstract** 4](#_Toc126153)

[**1.** **Introduction** 5](#_Toc126154)

[**1.1** **Background** 5](#_Toc126155)

[**1.2** **Purpose** 6](#_Toc126156)

[**1.2.1** **Project** 6](#_Toc126157)

[**1.2.2** **Report** 6](#_Toc126158)

[**1.3** **Scope** 7](#_Toc126159)

[**1.4** **Methodology** 8](#_Toc126160)

[**2.** **Problem Recognition** 11](#_Toc126161)

[**3.** **Feasibility Study** 11](#_Toc126162)

[**4.** **Analysis and User Requirements** 12](#_Toc126163)

[**5.** **Design** 13](#_Toc126164)

[**6.** **Implementation** 14](#_Toc126165)

[**7. Evaluation** 17](#_Toc126166)

[**8. Recommendations** 17](#_Toc126167)

[**9. Conclusion** 18](#_Toc126168)

[**10.** **Appendices** 19](#_Toc126169)

[**11.** **User Guide** 20](#_Toc126170)

[**11.1** **Connecting the MSP430 to the Raspberry Pi** 20](#_Toc126159)

[**11.2** **Executing the software code** 21](#_Toc126159)

# **Abstract**

The main objective of the Major Project is to simulate transferring of encrypted data from an Intelligence Agent to a street lamp and vice versa.

This will be done through generation of public and private keys by both parties. The public keys will be exchange with each other to generate a shared key with their own private key using the Diffie Hellman (DH) Key Exchange algorithm.

The initial key exchange algorithm idea was to use Elliptical Curve Diffie Hellman (ECDH) algorithm, but due to hardware limitations, time constraints and limited resources, it was not feasible at the moment. Hence, after 7 weeks into the project, I was advised to implement the standard Diffie Hellman (DH) algorithm on the devices.

The generated shared key will then be used to encrypt and decrypt data using Advance Standard Encryption (AES) method. In addition, only the receiver, who has the same shared key as the sender, is able to decrypt and read the data.

This simulation is being done using MSP430FR5969 board and Raspberry Pi, where the codes will be compiled into which will then be used to transmit and receive data through serial communication via Universal Asynchronous receiver-transmitter (Uart).

In conclusion, I was able to successfully complete the task to provide secure serial communications between two devices, namely an MSP430 board and Raspberry Pi.

1. **Introduction**

## **Background**

This report was written to document the work done for the Major Project entitled Smart Lighting. I was assigned as a one-man team with Dr Yap Chern Nam as the supervisor to complete the project during the span of 10 weeks. This project is coming to an end after being worked on for more than two years, and I was tasked to complete the project. I had to integrate my work with the previous groups’ work in order to produce Smart Lighting.

The previous group (AY2017/2018) was assigned to establish a serial communication over uart interface on MSP430 and Raspberry Pi via AES encryption and Elliptic Curve Cryptography. This is to ensure that the data being transmitted over from the Intelligence Agent to the lamp is secure. However, due to device and time constraints, they were unable to complete Elliptic Curve Cryptography on MSP430 side.

Hence, I have been tasked to implement cryptography algorithm and security for MSP430 and Raspberry Pi.

I have been tasked to complete the following parts of the project:

* Verify and validate the serial communication via UART on MSP430 and Raspberry Pi
  + Ensure that the data can always be sent and received.
* Implement Diffie Hellman (DH) algorithm codes, integrated with AES, in C/Python to ensure that data transmission from Intelligence Agent and Lamp is secure.
  + Ensure that the encrypted data is not changed and/or modified.
* Encryption and Decryption can always be done easily and successfully.

## **Purpose**

### **Project**

The objectives of the Major Project are:

* Transfer encrypted data from Intelligence Agent to the Street Lamp and vice-versa
* Generate both Public and Private keys for both Intelligence Agent and Street Lamp
* Generate Shared key for encrypting and decrypting data
* Exchange of Public keys to allow the intended parties to do the Diffie Hellman (DH) key exchange algorithm
* Encrypt the messages sent to each other with AES encryption
* Ensure that both sides is able to decrypt the encrypted message successfully

### **Report**

This report documents the research, contents, design, layout, implementation and testing work done for the Major Project. It also incorporates technical and non-technical recommendations to improve the project for future development, reference and archival work. The sources provided can be referred to for future use.

## **Scope**

The scope of the project is to ensure that the project exist and is able to perform its required task successfully. The project consists of the following to:

* Use UART for Serial Communication between Raspberry Pi and MSP430FR5969
* Using Diffie Hellman as key agreement algorithm
* Encrypt and decrypt data using Advanced Encryption Standard (AES) as Cipher Algorithm

Although the project was successfully completed, I faced numerous issues every week and it hindered my progress tremendously.

The largest major setback I had was that the initial key exchange algorithm idea was to use Elliptical Curve Diffie Hellman (ECDH) algorithm, but due to hardware limitations, time constraints and limited resources, it was not feasible at the moment. Hence, after 7 weeks into the project, I was advised to implement the standard Diffie Hellman (DH) algorithm on the devices.

In addition, there were little resources online that I could work with to try and solve the errors. For MSP430FR5969, there are hardware constraints, whereby it has a 16-bit architecture, low memory and storage. This make it not possible to utilize any publicly available library on Texas Instruments’ Code Composer Studio Compiler for Curve25519. Curve25519 produce a 256 bits key as such I can only use AES-256 as it accepts a 256 bits key. AES only encrypt and decrypt data in a 16 bytes block, as such there was a need to pad data with null byte(s) to make it a 16-byte block for encryption. Additionally, there is a delay when MSP430 read in the data as it will go through an interrupt to confirm that a data is being parsed before receiving it.

The computing limitation also contributed largely to the issues of this project. For example, in C code, the type “uint64\_t” is the largest type and could only hold up to a 20-digit long, 256-bit integer number. The largest 256-bit integer is **18,446,744,073,709,551,615. An integer larger than this number, would cause an error and the calculation in the code would fail. This led to many complications and resulted in failures of many implemented ideas and solutions for this project.**

## **Methodology**

In addition to the sources referenced by the previous groups, these are the sources that I have used throughout the duration of our project and they are mainly obtainable from the internet. The research materials are as follows:

|  |
| --- |
| **Getting Started with MSP430:** |
| <https://www.youtube.com/watch?v=3fO7PRcjXHE> |
| <https://www.youtube.com/watch?v=VW1LhWJd3UM> |
| <https://hacktronics.co.in/wtblog/1_blink-led-demo-code-for-msp430fr5969-launchpa.html> |
| <https://github.com/jck/msp430-dev-box/blob/master/examples/msp430fr5969/blink.c> |
| <https://gist.github.com/trtg/4451982> |

|  |
| --- |
| **Understanding DH and ECDH:** |
| <https://visualgdb.com/tutorials/msp430/> |
| <https://www.google.com/url?q=https://forum.43oh.com/topic/6432-can-i-program-a-msp430-using-microsoft-visual-studio/&usg=AFQjCNEM_cOAZMvnrtZTJ_KE4Fjrn62yvg> |
| <https://www.codeproject.com/Articles/1219531/Implement-Diffie-Hellman-in-Csharp> |

|  |
| --- |
| **ECDH, C coding Research Materials:** |
| <https://www.youtube.com/watch?v=F3zzNa42-tQ> |
| <https://munacl.cryptojedi.org/curve25519-msp430.shtml> |
| <https://ianix.com/pub/curve25519-deployment.html> |
| <https://www.google.com/search?q=MSP430+GNU+toolchain> |
| <http://www.ti.com/tool/MSP430-GCC-OPENSOURCE> |
| <https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/217067?SLAC460-link-is-broken> |
| <https://www.youtube.com/results?search_query=Curve25519+for+TI+MSP430> |
| <https://www.scribd.com/document/274966566/How-to-Convert-a-Project-From-IAR-to-CCS-Texas-Instruments-Wiki> |
| <https://en.wikipedia.org/wiki/TI_MSP430> |
| <https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/458065?How-to-use-IAR-to-CCS-ASM-perl-Script-> |
| <http://docplayer.net/50321304-Contents-of-msp430f21x2-code-examples-slac163-zip-asm-ccs-s43-iar-and-c-ccs-iar-link-to-zip-file.html> |
| <https://groups.google.com/forum/#!topic/comp.arch.embedded/e8rOUQ9G9vo> |
| <https://www.embedded.fm/blog/ese101-assembly-language-playground> |
| <https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/458065?How-to-use-IAR-to-CCS-ASM-perl-Script-> |
| <https://courses.cs.washington.edu/courses/cse466/13au/pdfs/ccsv42slau157s.pdf> |
| <https://www.cs.auckland.ac.nz/references/unix/digital/AQTLTBTE/DOCU_078.HTM> |
| <https://e2e.ti.com/support/microcontrollers/msp430/f/166/t/122956?how-to-define-a-bit-datatype-in-IAR-for-MSP430-> |
| <http://mhutter.org/research/msp430/> |
| <https://cryptojedi.org/papers/mspeed-20141001.pdf> |
| <http://discovery.csc.ncsu.edu/software/TinyECC/TR-2007-36.pdf> |
| <https://orbilu.uni.lu/bitstream/10993/25601/1/ASIACCS2015.pdf> |
| <https://link.springer.com/chapter/10.1007/978-3-642-10628-6_17> |
| <https://www.swmath.org/software/9089> |
| <https://www.reddit.com/comments/2q2zbv> |
| <https://www.embeddedrelated.com/showthread/msp430/45008-1.php> |
| <https://cryptojedi.org/papers/mspeed-20141001.pdf> |

|  |
| --- |
| **Simple Diffie Hellman Research:** |
| <https://sublimerobots.com/2015/01/simple-diffie-hellman-example-python/> |
| <https://stackoverflow.com/questions/43886464/cryptography-python-diffie-hellman-key-exchange-implementation> |
| <https://null-byte.wonderhowto.com/how-to/generate-private-encryption-keys-with-diffie-hellman-key-exchange-0180269/> |
| <http://dandylife.net/blog/archives/295> |
| <https://pypi.org/project/python-pkcs11/> |
| <http://codingloverlavi.blogspot.com/2015/04/diffie-hellman-key-exchange-algorithm.html> |
| <https://www.geeksforgeeks.org/implementation-diffie-hellman-algorithm/> |
| <https://brilliant.org/wiki/diffie-hellman-protocol/> |
| <https://slideplayer.com/slide/5789157/19/images/9/Diffie-Hellman+Mathematics.jpg> |
| <https://security.stackexchange.com/questions/45963/diffie-hellman-key-exchange-in-plain-english> |

|  |
| --- |
| **GitHub and Coding Research Materials:** |
| <https://github.com/miracl/MIRACL/blob/master/source/msp430.mcs> |
| <https://www.embeddedrelated.com/showthread/msp430/45008-1.php> |
| <https://electronics.stackexchange.com/questions/144343/send-integer-values-through-serial-communication-in-pic-to-pic> |
| <https://www.embeddedrelated.com/showarticle/420.php> |
| <https://www.badprog.com/c-type-converting-two-uint8-t-words-into-one-of-uint16-t-and-one-of-uint16-t-into-two-of-uint8-t> |
| <https://www.raspberrypi.org/forums/viewtopic.php?t=9025> |
| <https://www.tutorialspoint.com/c_standard_library/c_function_memcpy.htm> |
| <https://visualgdb.com/tutorials/msp430/uart/> |
| <https://www.tutorialspoint.com/c_standard_library/c_function_memcpy.htm> |
| <http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/122103> |
| <http://e2e.ti.com/support/microcontrollers/msp430/f/166/t/720340?CCS-MSP430FR5969-Embedded-ECC-solutions-> |
| <https://github.com/agl/curve25519-donna> |
| <https://github.com/silvaCiminia/py_ECDHE> |
| <https://github.com/pyca/cryptography/blob/master/docs/hazmat/primitives/asymmetric/ec.rst> |
| <https://github.com/j2kun/elliptic-curve-diffie-hellman/blob/master/elliptic.py> |
| <https://cryptography.io/en/latest/hazmat/primitives/asymmetric/dh/> |

|  |
| --- |
| **AES Research Materials:** |
| <https://www.iar.com/support/resources/articles/secure-mastering/> |
| <https://crypto.stackexchange.com/questions/9509/aes-encryption-using-a-diffie-hellman-key-exchange/9510> |
| <https://sites.google.com/site/mobilesecuritylabware/3-data-location-privacy/lab-activity/cryptography/cryptography-mobile-labs/key-management/km-post-lab/assign1> |

1. **Problem Recognition**

My supervisor has updated me on the current progress of the project that has been ongoing fortwo years. For the Smart Lighting currently, encryption on and communication between the Raspberry Pi and MSP430FR5969 is implemented by the previous group of AY17/18. As such Raspberry Pi and MSP430FR5969 will be able to communicate with each other through serial connection.

However, the serial communication is without encryption, hence communication will not be secure.

1. **Feasibility Study**

Although there are many different aspects of feasibility studies. This project will be focussing on the economic and technical aspects.

Economic feasibility assesses the cost required for the different phases of the project such as researching, developing and implementing Smart Lighting in future. An analysis is being done to determine if the product is cost efficient by comparing the total costs and the expected benefits of the project. The development cost will include resources to aid in the research process, equipment that are necessary for this project. Additionally, a group of specialists may be required in future to further improve the project. This project requires devices such as Raspberry Pi and MSP430 boards. Therefore, a substantial amount is needed to be set aside to purchase the desired number of devices. Besides, if the project were to be deployed vastly in the future, expertise will be essential for upgrading the codes and security implementations.

The other feasibility aspect for our project would be technical aspect. This suggests whether the technical team will be able to implement the project with the available resources provided.  Additionally, it is also essential to evaluate the software and hardware requirements for the project. As this project involves hardware, some software may be incompatible due to the restrictions of the hardware. Therefore, the technical analysis must be considered to ensure that the project will run smoothly.

1. **Analysis and User Requirements**

I am required to assess the user's needs and expectations for our potential clients which could be the government or a company that has plans on deploying Smart Lighting in the future. The user requirements that have been discussed are as follows:

* User is able to control the lamp remotely by switching on and off the lights
* User is able to create shared key easily
* User is able to encrypt and decrypt data easily
* User is able to obtain the devices easily from the market

It is important to have a clear understanding of the user requirements which will aid in the development of the project by addressing to the user’s needs.

1. **Design**

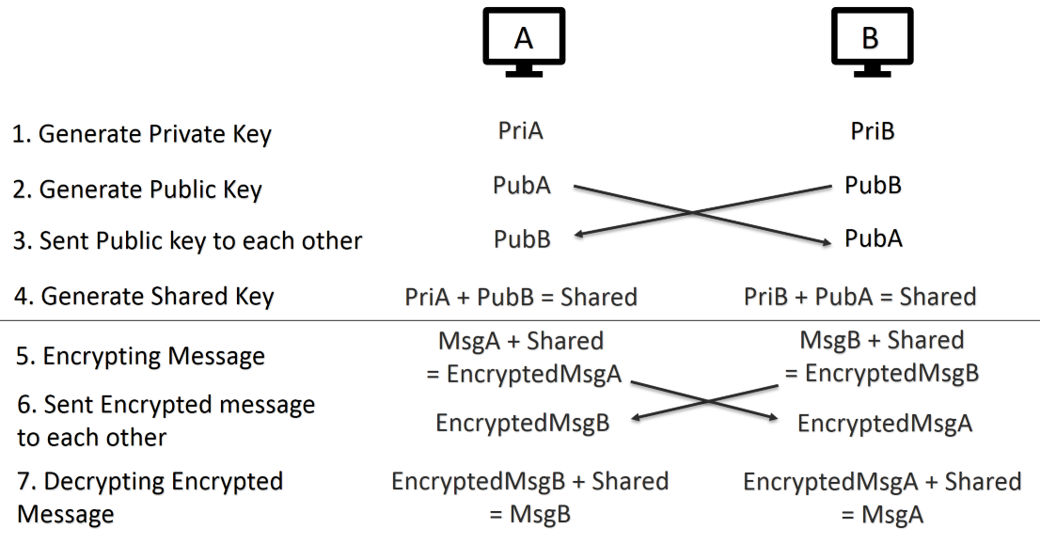
* **Aims**
  + Encrypted communication between MSP430FR5969 and Raspberry Pi
  + Advanced Encryption Standard (AES) cipher for encryption and decryption.
  + Universal Asynchronous receiver-transmitter (UART) for communication.
  + Diffie-Hellman for key agreement.
* **Rationale**
  + Secure Communication between Raspberry Pi, as main controller, and MSP430FR5969, as controller for light. This is to prevent unauthorised access and control of Raspberry Pi and MSP430FR5969.
* **Layout**
  + Diffie Hellman (DH) and Advanced Encryption Standard (AES) will be embedded on MSP430 (E.g. Lights).
  + Raspberry Pi will also have same features Diffie Hellman (DH) and Advanced Encryption Standard (AES) but Raspberry Pi will be the controller to control the lights.

1. **Implementation**

For encrypted communication between MSP430 and Raspberry Pi, it consists of a few components. Firstly, to communicate between these 2 devices, I will be using Universal Asynchronous receiver-transmitter (UART). Secondly, is to encrypt the data to be sent over to each other, It will consist of Diffie Hellman (DH) as the key agreement algorithm; Advanced Encryption Standard (AES) as the cipher algorithm. The algorithm would generate a 256 bits key and will need to utilize AES-256 as it accepts a 256 bits key. As AES-256 only encrypt in a 16 bytes block, there will be the need to pad with null byte if the data size modulus 16 is greater than 0.

For MSP430 (represented by A) and Raspberry Pi (represented by B):

* 1. Both will need to generate their own secret key, this is done by generating a random secret key.
  2. Using the biggest 64 bit Prime, P = 0xffffffffffffffc5ull; and the Generator, G = 5, both parties will generate their public keys.
  3. Both will exchange their public keys via UART
  4. With their own secret key and the other’s public key, each party will generate a shared key. This shared key generated on both sides will be the same.
  5. If the message to be encrypted modulus 16 is greater than 0, it will be padded with null byte(s) till the padded message modulus 16 equal to 0, the padded data will be known as data. If the data to be encrypted modulus 16 is equal to 0, the message will be known as data. To encrypt, both will use AES-256 with their own data and shared key as input, it will generate a ciphertext, which will be known as the encrypted data.
  6. Both sides will exchange their encrypted data to the other for decryption.
  7. To decrypt, both will use AES-256 with the other encrypted data and shared key as input, this will output the other decrypted data.



MSP430 which will be programmed in C and Texas Instruments’ Code Composer Studio Assembly Language, in Code Composer Studio. Raspberry Pi will be programmed in Python.

Due to some complications, there are some additional features with regards to the implementation design as stated above.

* + 1. Firstly, the type, “uint64\_t” is unable to hold the incredibly large integers needed for calculation and the results of the calculation, hence the generated shared key size has a threshold and the process of generating the shared key must be repeated until the size of the shared key is big enough.
    2. To maintain communication stability, the first few exchanges of data communication between the two parties via serial communication is to establish the connection and to eliminate any stray leftover data being held at the communication ports.
    3. There was the need to convert large integers to characters, ascii and vice versa. This meant separating the strings into characters or integers into digits (sometimes in pairs and in proper order), converting each individual characters or digits, and concatenating them into strings and integers together. A function to reverse the order of the string is needed as well.

Every function in the programme are tested repeatedly and verified with external sources to ensure that the function was programmed to do exactly what it was intended to do.

For example, the conversion of integer to ASCII and the encryption results are verified using an online ASCII Converter website. (<https://www.branah.com/ascii-converter>)

# **7. Evaluation**

From the codes provided by the previous group, together with my codes, there is a delay reading from MSP430 via serial communications, hence there was a need to loop the data sent by Raspberry Pi to MSP430 for MSP430 to receive it successfully.

Previously, there is also additional byte in the form of the 16-byte encrypted data send from Raspberry Pi to MSP430, as the encrypted data of Raspberry Pi is 2nd to 17th byte when viewed from MSP430. But this was managed and evaded from the process of having the start of the first few exchanges of data communication eliminate any stray leftover data being held at the communication ports.

# **8. Recommendations**

1. Implementing Cipher Block Chaining (CBC) mode for AES, as it is more secure as compare to the current mode of AES which is Electronic Codebook (ECB).
2. Implement expiry for secret, private and shared key after a certain period, to reduce CPU load due to calculation of key, as MSP430 microcontroller is not powerful, generating a new set of keys for each communication will reduce performance.
3. Implementing time synchronisation to reduce time drift among devices, which will impact time sensitive tasks.
4. Implement Elliptic-Curve Diffie-Hellman (ECDH), a variant of the simple Diffie Hellman (DH) algorithm, which is much more efficient and secure.

**9. Conclusion**

The duration of this project exceeded the 10 weeks schedule, as anticipated due to the difficulty of the project.

There was great difficulty in finding resources and libraries to be use on MSP430 microcontrollers coupled with hardware constraint for MSP430, MSP430 is 16-bit architecture, low performance, low memory and low storage processor.

The hardware constraint resulted in not being able to utilize any of publicly available library for Curve25519 Elliptic-Curve such as curve25519-donna or Networking and Cryptography library (NaCl). We can’t use an assembly language program we found that has been tested to work as it is in another Assembly Language, GNU Compiler Collection (GCC), which is not supported by Texas Instruments’ (TI) Code Composer Studio (CCS) Assembly Language.

TI’s CCS Assembly Language for MSP430’s resources are hard to find as compared to other more popular assembly language such as Arduino or GCC. With this new hurdle, I must understand how Curve25519 Elliptic-Curve work in code in order to implement it for MSP430.

With more time, I’m confident that the mode of AES used can be improved by using CBC mode instead of ECB mode and possibly implement Elliptic-Curve Diffie-Hellman instead of simple Diffie Hellman algorithm, as well as implementing a shared key’s expiration date.

From this project, I have learnt how hardware constraints shape coding designs, how mathematical algorithms and formulas are translated into code, how elliptic curves has evolved from being a math formula to a being used in a robust and secure cryptography algorithm. Additionally, I learnt in depth about how AES works, the difference between the AES modes and the theory behind the encryption steps.

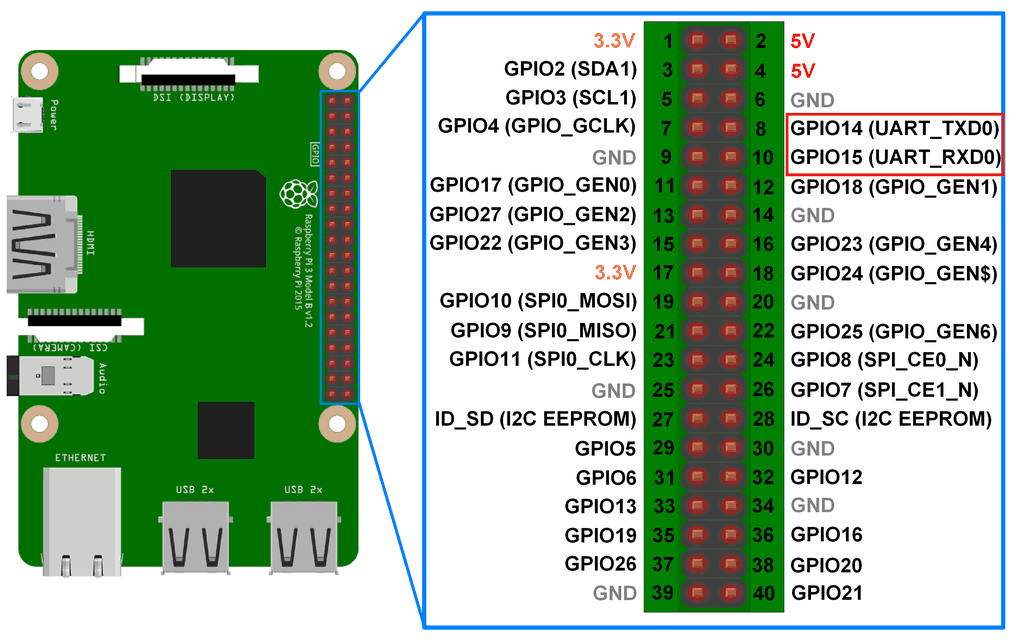
# **Appendices**



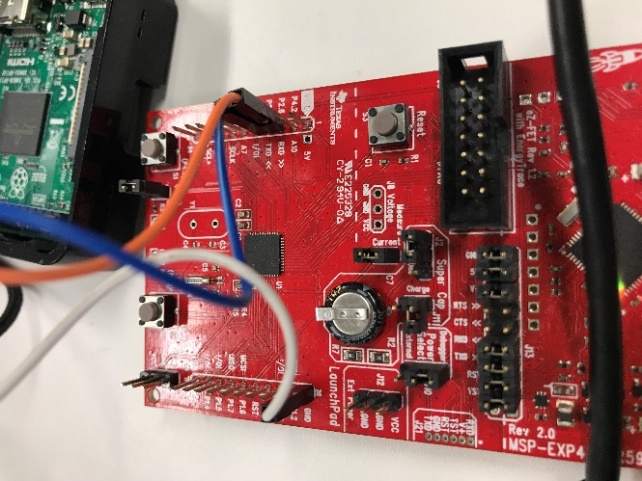
# **User Guide**

* 1. **Connecting the MSP430 to the Raspberry Pi:**

1. Connect the ground wire to pin 6 on the Raspberry Pi
2. Connect the Tx wire to pin 8 on the Raspberry Pi
3. Connect the Rx wire to pin 10 on the Raspberry Pi



An example is shown in the following images:



* 1. **Executing the software code:**

1. Unzip **"DH uart test4(Cipher working, AES Blend)"** file into

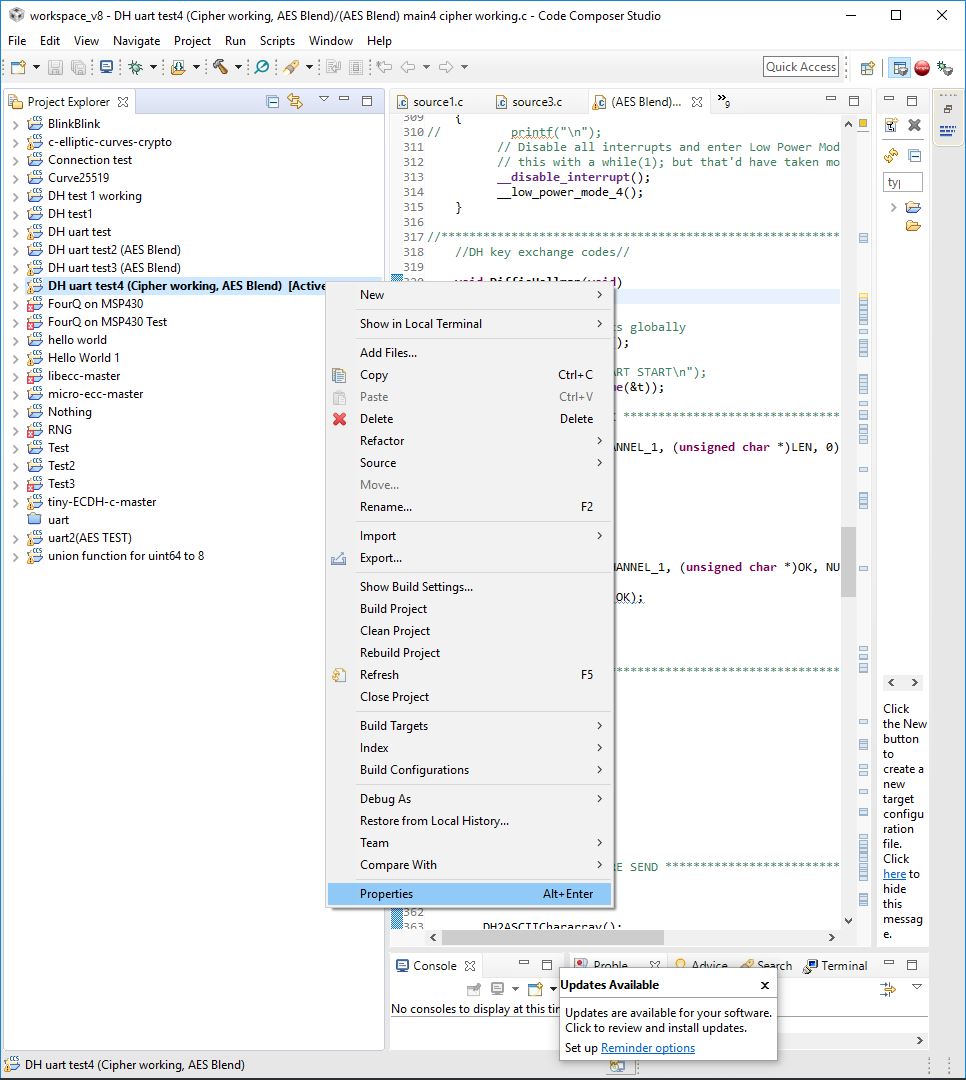


**“C:\Users\User\workspace\_v8"**

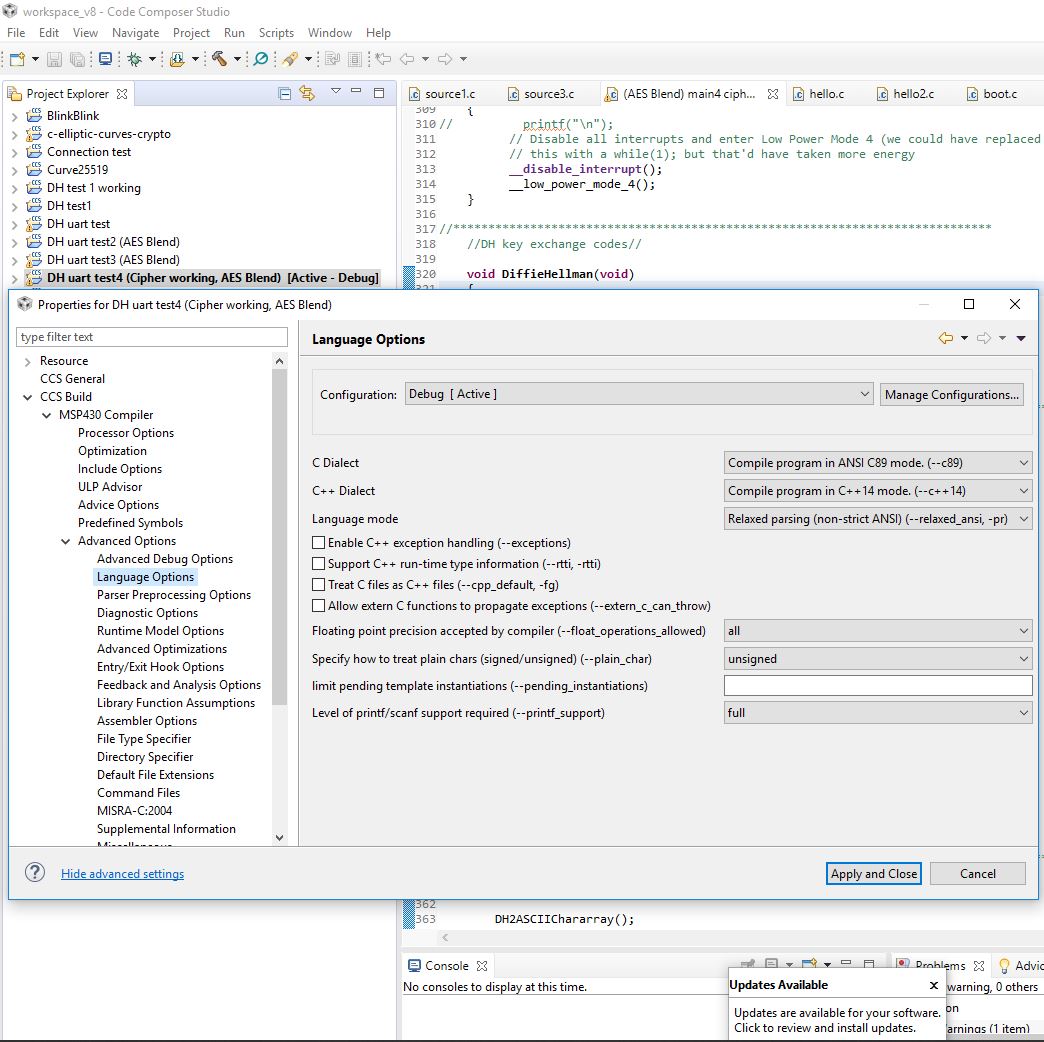
Or wherever Code Composer Studio’s (CCS's) workspace\_v8 is found in your drive.

* 1. In CSS:

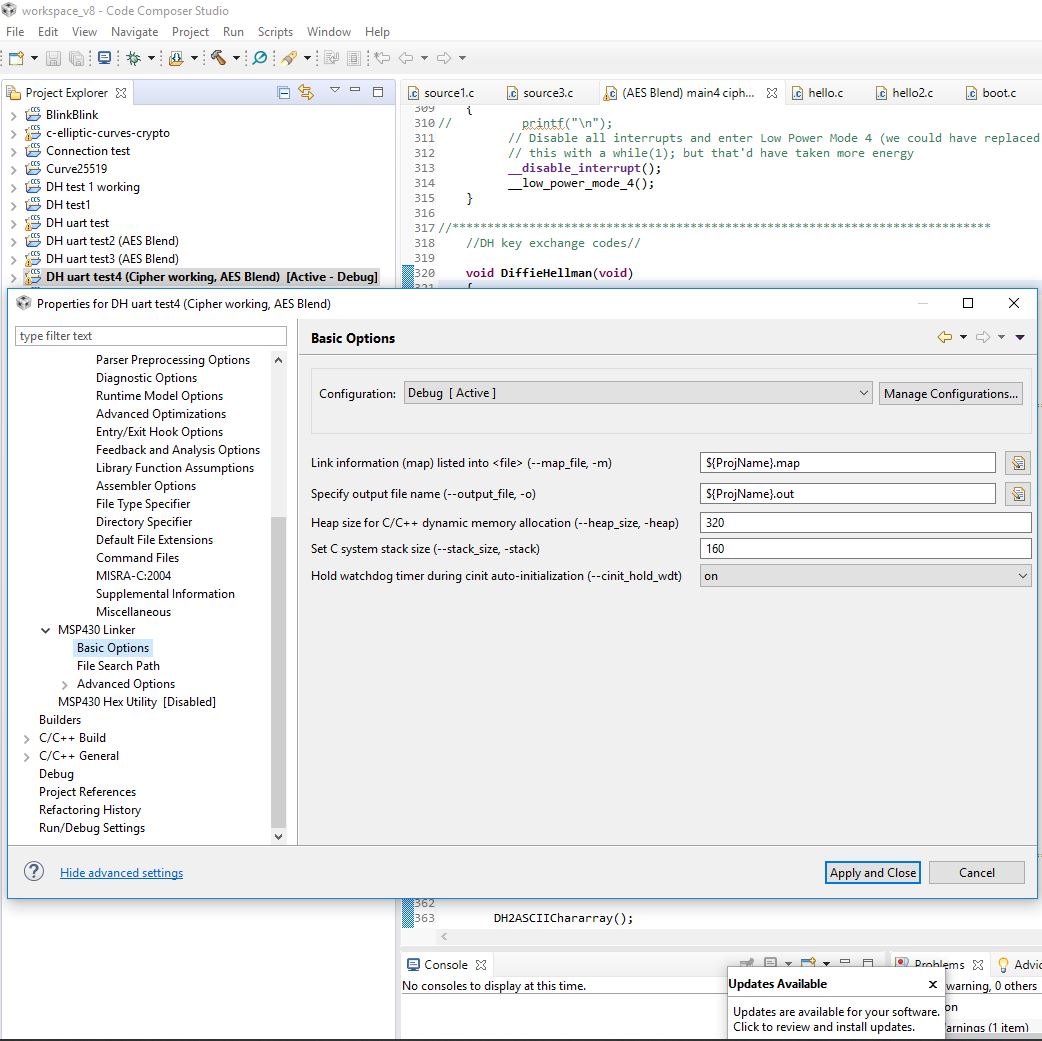
Right click on the CSS project and select properties:



* 1. Properties > MSP 430 Compiler > Advanced Options > Language Options > "Level of printf/scanf support required (--printf\_support) = "full".



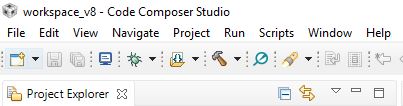
* 1. Properties > MSP 430 Compiler > MSP430 Linker > Basic Options > "Heap size for C/C++ dynamic memory allocation (--heap\_size,-heap) = "320"



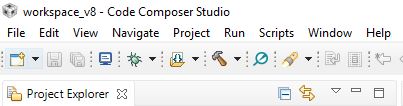
1. Download **"DH python (AES Blend).py"** file into your **Raspberry Pi.**



1. To execute the project:
   1. Open the **"DH python (AES Blend).py"** file in your **Raspberry Pi** and press F5 to run it.
   2. Then start the script in **CCS** by building the script by **clicking on the hammer button**:



* 1. Then debug it by **clicking on the green bug button**:



* 1. Then finally, starting the script by **clicking on the play button**: