

eYSIP2016

ROBOT STATE COLLECTOR



Amanpreet Singh

Amit Raushan

Shubham Gupta

Duration of Internship: 10/06/2016 – 24/07/2016

2016, e-Yantra Publication

Contents

1	Robot State Collector	2
1.1	Abstract	2
1.2	Completion Status	3
1.3	Hardware Parts Used	4
1.4	Software Used	12
1.5	Assembly of Hardware	13
1.5.1	Fire Bird V Robot	13
1.5.2	Tiva Robot	13
1.5.3	Steps for Assembling Tiva Robot	17
1.6	Software and Code	23
1.6.1	State Collection Code for Fire Bird V Robot	23
1.6.2	Client GUI and Server	23
1.7	Use and Demo	24
1.7.1	Steps for merging the State Collection Code with User's Code for Fire Bird V	24
1.7.2	Steps for merging the State Collection Code with User's Code for Tiva Robot	24
1.7.3	Steps for Using the GUI	24
1.7.4	Steps for setting up the Server	25
1.8	Future Work	26
1.9	Bug Report and Challenges	27
1.9.1	Bugs	27
1.9.2	Challenges	27

Robot State Collector

1.1 Abstract

This project aims at developing a State Collection Algorithm which is easy to integrate with user code without interfering with it. Our code should be capable of collecting the sensor values, the buzzer and LCD data periodically. It should also send this data to the GUI developed for this specific purpose. The GUI should be able to encrypt the collected data and send it to the e-Yantra servers.

This would be helpful in virtually re-creating runs ,using the user's code and the data collected by us.

The main tasks involved in the project are :

1. Collecting timed data about the state of the robot(sensors etc.).
2. Developing a GUI that picks up the data from the robot,encrypts it and send it to e-yantra servers.
3. Writing server side code to receive and decrypt the state information obtained from GUI developed in step 2.



1.2 Completion Status

1. Successfully collected the data on Fire Bird V and Tiva Robot using the specially developed State Collection Algorithm.
2. To avoid storing the data on the volatile memory on any robot a XBee module is used which directly sends the data to the GUI wirelessly. This also saves space on the robot's memory.
3. Developed a GUI to read the incoming data from the Robot. The GUI encrypts the data and sends it back to the server. The GUI is also capable of detecting if any changes were made to the file before sending it to the server.
4. Made a Robot using the Tiva board so as to check that our algorithm and GUI are not restricted to only Fire Bird V.
5. Performed state collection on the newly developed robot.

1.3 Hardware Parts Used

- Hardware used with Fire Bird V :

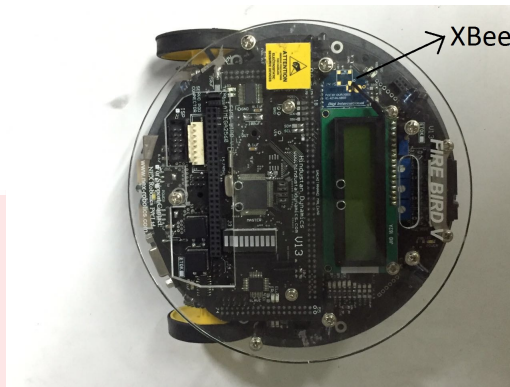


Figure 1.1: Fire Bird V with XBee module

1. XBee Module.
[Datasheet](#)



Figure 1.2: XBee module

1.3. HARDWARE PARTS USED

2. XBee Module and adapter to receive data on a Laptop/PC.

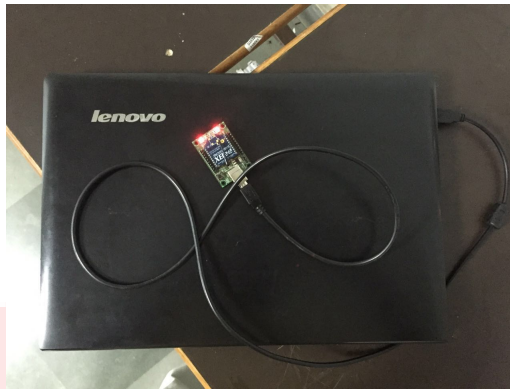


Figure 1.3: XBee with adapter connected to laptop

- Hardware used to build a Robot using Tiva board (TM4C123GXL)
[Datasheet](#)
[Peripheral Driver Library](#)
[User's Guide](#)

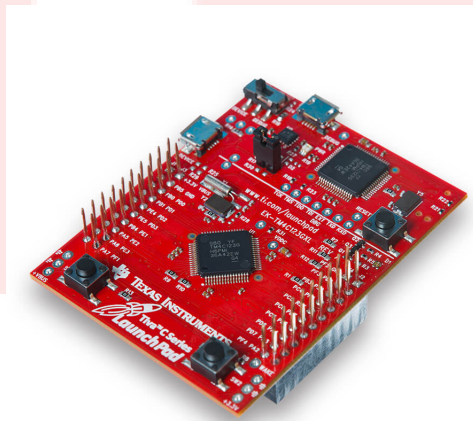


Figure 1.4: Tiva C Series TM4C123G LaunchPad

1.3. HARDWARE PARTS USED

1. 2x DG02S Mini DC Gear motor.

[Datasheet](#)



Figure 1.5: DC Geared motor

2. 2x Wheel - 65mm in Diameter.



Figure 1.6: Wheel

3. L293D - Motor Driver.

[Datasheet](#)

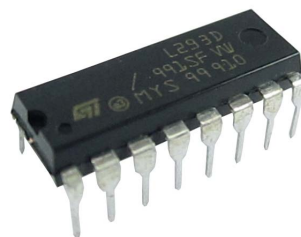


Figure 1.7: Motor Driver IC

1.3. HARDWARE PARTS USED

4. 16x2 LCD.

[Datasheet](#)

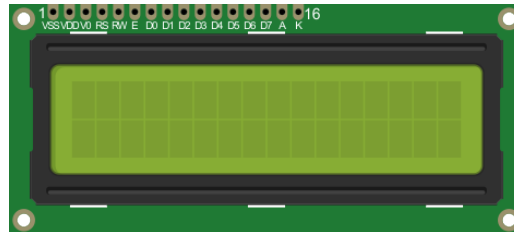


Figure 1.8: LCD 16x2 Display

5. White Line Sensor Module.

[Manual](#)



Figure 1.9: 3 Channel Whiteline Sensors

6. SHARP 0A41SK sensor.

[Datasheet](#)



Figure 1.10: SHARP Sensor

1.3. HARDWARE PARTS USED

7. Caster Wheel.



Figure 1.11: Caster Wheel

8. LM3237 - Voltage Regulator.

[Datasheet](#)

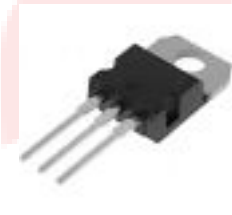


Figure 1.12: LM3237

9. Heat Sink.

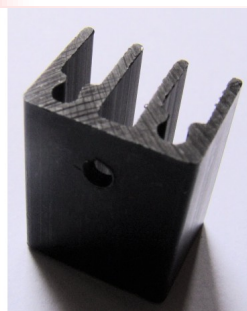


Figure 1.13: Heat Sink

1.3. HARDWARE PARTS USED

10. 10 μ F Electrolyte Capacitor.

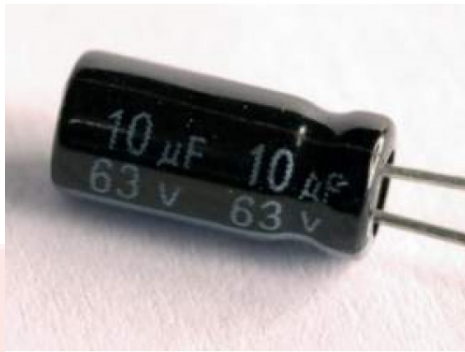


Figure 1.14: Capacitor

11. Multi-purpose PCB board.

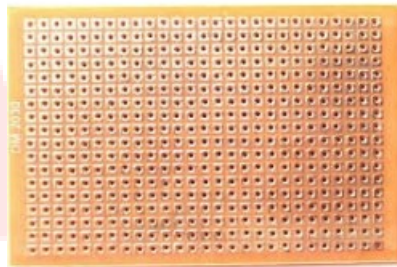


Figure 1.15: PCB Board

1.3. HARDWARE PARTS USED

12. 12V Rechargeable Battery.

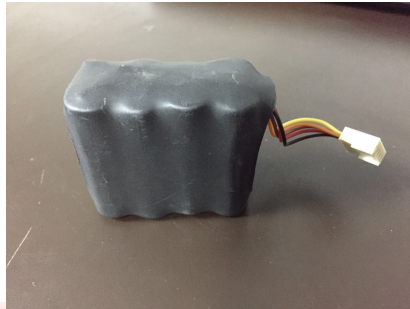


Figure 1.16: 12V Rechargeable Battery

13. 20 Pin Planar Cable.



Figure 1.17: 20 pin Connector wire

14. Female bug strip.



Figure 1.18: Female bug strip

1.3. HARDWARE PARTS USED

15. Male bug strip.

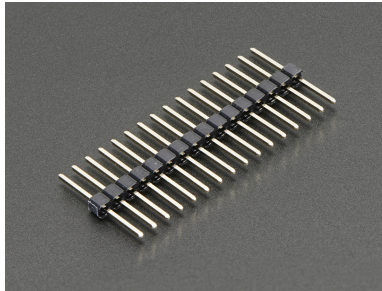


Figure 1.19: Male bug strip

16. Male to Female Jumper Wires.



Figure 1.20: Jumper Wires

17. Plastic Chassis.



Figure 1.21: Chassis



1.4 Software Used

- Atmel Studio 6.0
[Download link](#)
- XCTU-NG
[Download link](#)
- Serial Terminal
[Download link](#)
- Code Composer Studio v6.1.3
[Download link](#)
- NetBeans IDE 8.1
[Download link](#)
- AVR Bootloader
[Download link](#)



1.5 Assembly of Hardware

1.5.1 Fire Bird V Robot

XBee module is attached with Fire Bird V Robot to transmit the collected state information.

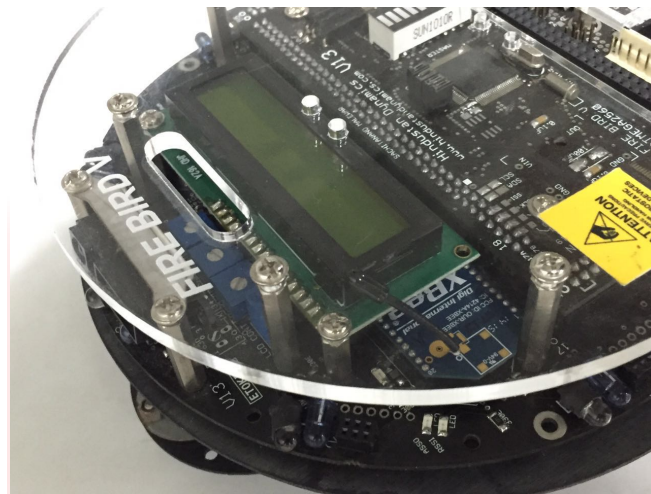


Figure 1.22: XBee connected with Fire Bird V

1.5.2 Tiva Robot

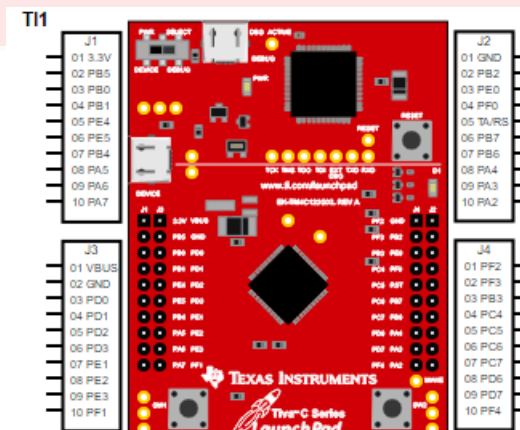


Figure 1.23: Tiva Board Pin Location



1.5. ASSEMBLY OF HARDWARE

Connections of PORT pins of Tiva board with different components:

1. Interfacing of white line sensor

- PE1 - LEFT SENSOR
- PE2 - CENTER SENSOR
- PE3 - RIGHT SENSOR
- GND - GND
- VCC - VCC

2. Interfacing of Sharp sensor

- PE0 - Sharp SENSOR
- GND - GND
- VCC - VCC

3. Interfacing of Motor Driver IC

- PE4 - PWM1
- PA2 - IN1
- PA3 - IN2
- GND - GND
- VCC - VCC
- PE5 - PWM2
- PA6 - IN3
- PA7 - IN4

4. Connecting the wheels to the Motor Driver IC.

- OUT1 , OUT2 - Left Wheel Motor
- OUT3 , OUT4 - Right Wheel Motor



1.5. ASSEMBLY OF HARDWARE

5. Interfacing of XBee module

- PC4 - Rx
- PC5 - Tx
- GND - GND
- VCC - VCC

6. Interfacing of Buzzer

- PF0 - Signal
- GND - GND

7. Interfacing of LCD

- PB0 - DATA 0
- PB1 - DATA 1
- PB2 - DATA 2
- PB3 - DATA 3
- PB4 - DATA 4
- PB5 - DATA 5
- PB6 - DATA 6
- PB7 - DATA 7
- GND - GND
- VCC - VCC
- VEE - GND
- VCC - Led+
- GND - Led-
- PA4 - RS
- PA5 - RW
- PC6 - EN

Block Diagram

Basic connections of different components used with Tiva Robot.

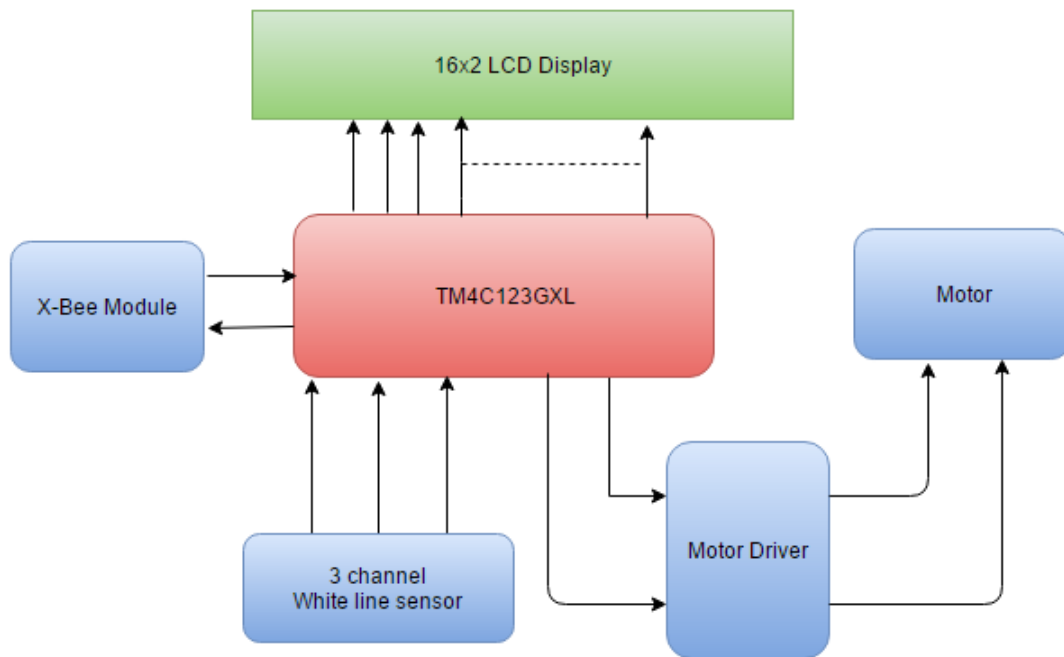


Figure 1.24: Block Diagram of Tiva Robot

1.5.3 Steps for Assembling Tiva Robot

1. Gathering/Buying all the components to be used for building the Robot.

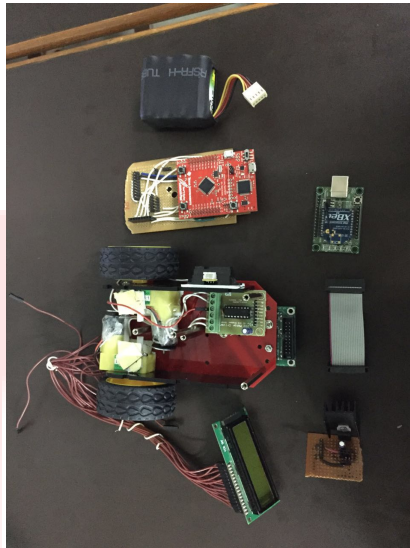


Figure 1.25: All the components used to build the Robot.

2. Assemble the chassis, two dc geared motors and two wheels. Then connect the motors with the Motor Driver IC.

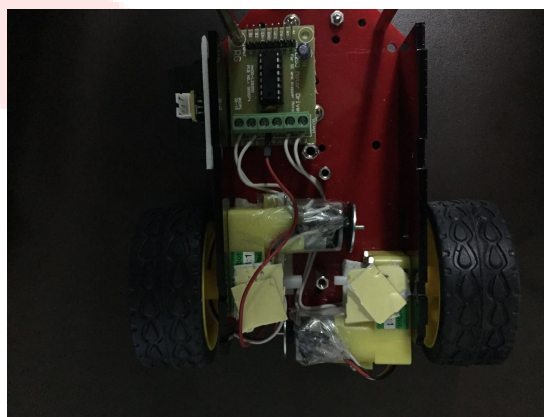


Figure 1.26: Motors and wheels attached to chassis

1.5. ASSEMBLY OF HARDWARE

3. Connect the White Line sensors and Caster wheel to the chassis.

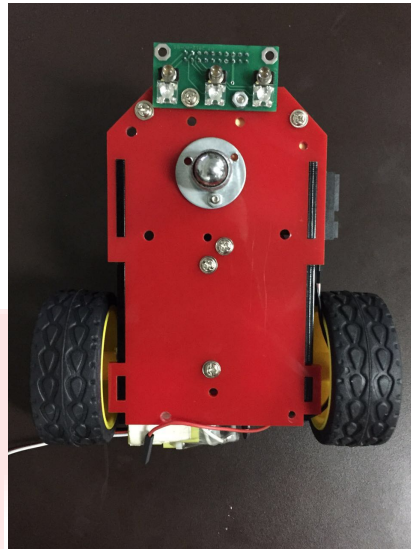


Figure 1.27: Caster Wheel and White Line Sensors attached to chassis

4. Attach SHARP Sensor to the chassis.

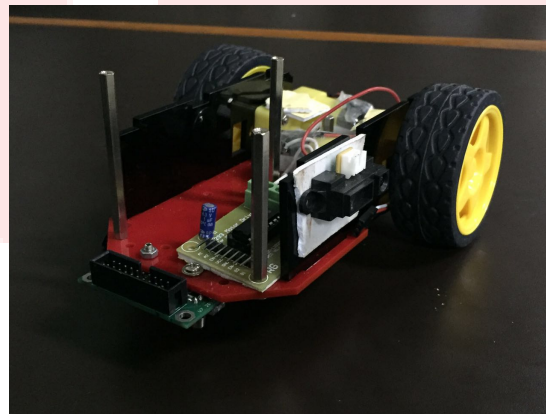


Figure 1.28: SHARP Sensor attached to chassis

1.5. ASSEMBLY OF HARDWARE

5. Attach the Buzzer to the PCB board. Using the male bug strips (4x10) design a PCB board to which the Tiva board will be attached.

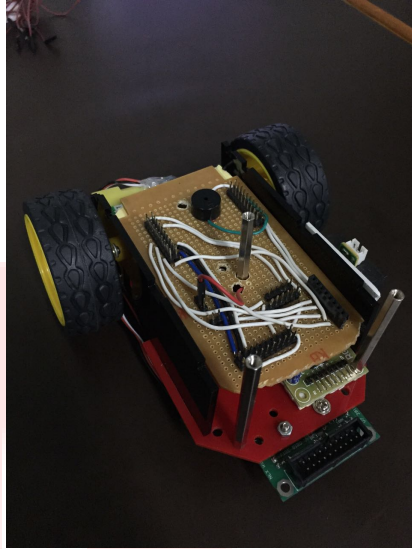


Figure 1.29: Buzzer which would be beneath the Tiva board

6. Attach Tiva board (TM4C123GXL) on the PCB board.

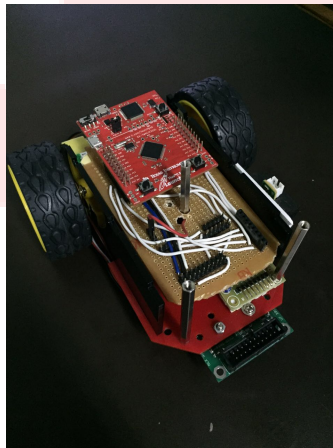


Figure 1.30: Tiva board fitted

1.5. ASSEMBLY OF HARDWARE

7. Connect White Line Sensors using the 20 pin connector.

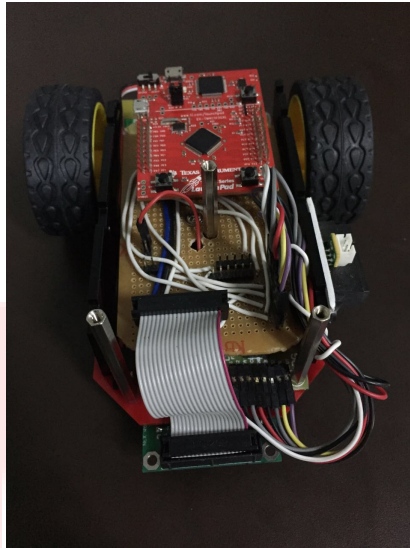


Figure 1.31: Whiteline Sensors are now connected to the Tiva board.

8. Connect SHARP Sensor and Motor Driver IC to the Tiva board.

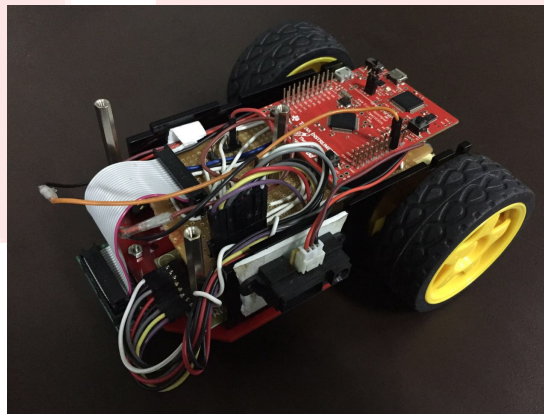


Figure 1.32: SHARP Sensor and Motor Driver IC are now connected to the Tiva board.

1.5. ASSEMBLY OF HARDWARE

9. Conenct the LCD to the Tiva board.

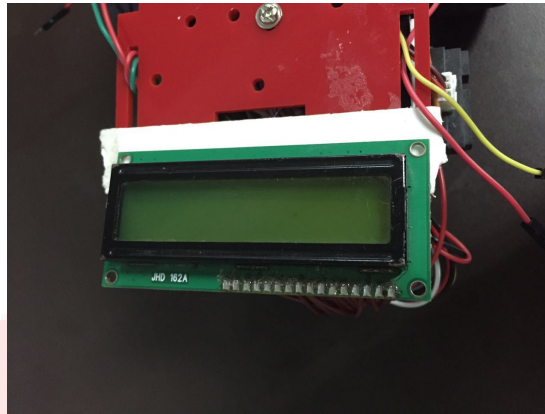


Figure 1.33: LCD after being attached to the Robot.

10. Connect the XBee to the Tiva board.

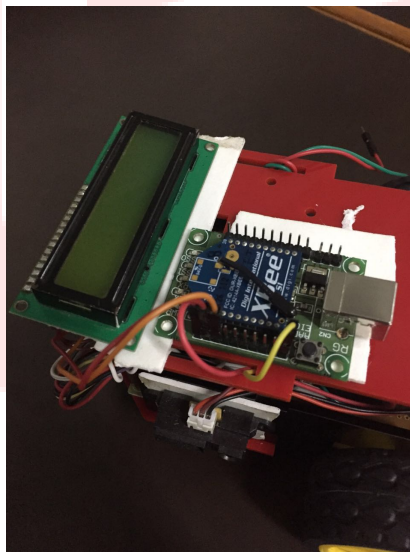


Figure 1.34: XBee after being attached to the Robot.

1.5. ASSEMBLY OF HARDWARE

11. Connect the Voltage Regulator Circuit and Battery to the Tiva board.

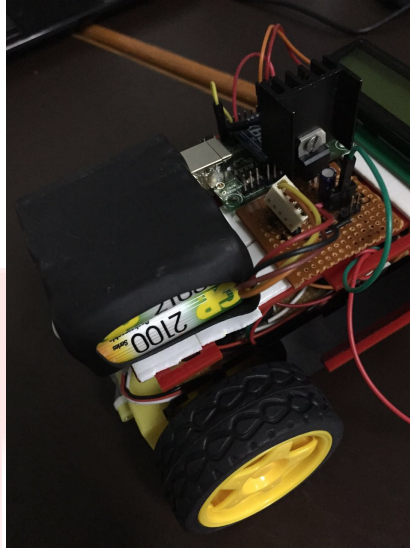


Figure 1.35: Voltage Regulator Circuit and Battery after being attached to the Robot

12. Now all the Components are attached to the Robot.

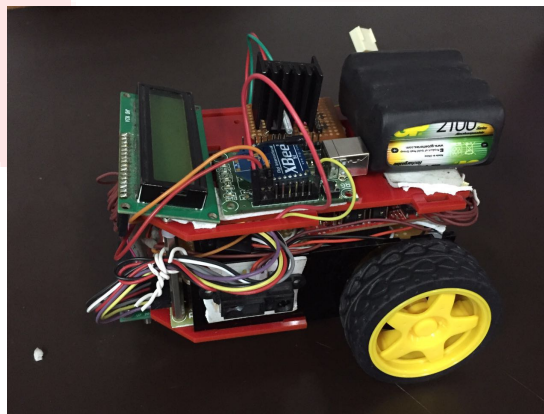


Figure 1.36: Tiva Robot.



1.6 Software and Code

[Github link](#) for the repository of code

1.6.1 State Collection Code for Fire Bird V Robot

We use a specific Timer to generate an interrupt after 0.5s. Whenever the interrupt occurs, the state information(i.e. values of all sensors etc.) is collected and transmitted wirelessly to the PC/Laptop via XBee.

Our code is entirely inside the provided header file. A simple function call starts the State Collection.

The specific timers used by us to collect the state data for different Robots are as follows:

- For Fire Bird V – Timer 4
- For Tiva Robot - Timer0A

NOTE: The Timer used by us cannot be used by the user.

1.6.2 Client GUI and Server

The GUI is capable of reading the incoming data from the XBee module. We use a symmetric key encryption scheme to encrypt the collected the state data. The key of the symmetric key encryption scheme is encrypted using a public key encryption scheme. At the end the encrypted data can be sent to the server if the user wishes.



1.7 Use and Demo

1.7.1 Steps for merging the State Collection Code with User's Code for Fire Bird V

1. include "state_collect.h" in User' Code
2. In main after initialising all the devices call the function "_init_devices()"

NOTE : The files included above should be present inside the same folder as that of the User's code.

1.7.2 Steps for merging the State Collection Code with User's Code for Tiva Robot

1. include "common_header.h" in User' Code
2. In main call the function "start_collection()".

NOTE : The files included above should be present inside the same folder as that of the User's code.

1.7.3 Steps for Using the GUI

1. Login with the credentials provided.
2. Make sure that the XBee adapter is connected to the laptop.
3. Click on the "Search for available COM ports " button.
4. From the Drop Down Menu select the COM port on which the XBee adapter is connected. Selecting the COM port would establish a connection with the XBee and the "Not connected" would change to "Connected".
5. After the run is complete click on "End of Run" button to indicate the end of the robot's task.
6. To send the data to the server click on "Connect to the Server" button.
7. To disconnect from the COM port press the "Disconnect" button.



1.7. USE AND DEMO

1.7.4 Steps for setting up the Server

1. Make a Java project and copy the contents of Server Side code, "decp.java", in the main class.
2. In the same folder as that of your project files listed below should be present.
 - "keyStoreName.jceks"
 - "ip.txt"
3. Now Run the Java program and the server is now started.
4. To generate a new Private and Public key Pair just run the code within "Generation of Private and Public Key.java". And then just provide the newly generated "publicKey.txt".



1.8. FUTURE WORK

1.8 Future Work

1. LCD data can be collected.
2. Position encoders can be configured with the Tiva Bot.





1.9 Bug Report and Challenges

1.9.1 Bugs

1. XBee doesn't run at baud rate of 115200 on Fire Bird V.
2. XBee doesn't run at baud rate of 9600 on Tiva bot.
3. If we change the frequency of the system clock on Tiva bot the XBee doesn't work.
4. If we have to collect data of the position encoders then it has to be done with the interrupt used by the user and hence we have to change the user's code a bit.
5. The server side code hasn't been tested with multiple clients.
6. The server would give an error if the user logs in and closes the GUI without sending the data.
7. We don't know what would happen if connection between the client and the server is lost while transmitting the data.
8. There would be a loss of data if the user presses the Disconnect button while the GUI is collecting the State data.
9. On Tiva board no formula for conversion of digital values of SHARP sensor to actual distance has been used as Tiva board has a 12 bit ADC channel and all the formulas we found were meant for 8 bit ADC channels.

1.9.2 Challenges

1. Storing the Data efficiently on the Robot's memory – This was avoided by sending the data using a XBee module.
2. We were first transmitting the data as soon as it was read. This caused a problem as the earlier data was not sent and the new data arrived which caused them to overlap and hence resulted in loss of information – This was solved by waiting for the previous data to be sent first and then sending the new data.



1.9. BUG REPORT AND CHALLENGES

3. Encryption without using any padding as the data size was not a multiple of 16 bytes and hence the AES encryption algorithm would fail – This was solved by using a padding which ensures that the data is always a multiple of 16 bytes.
4. Sending the data in a integer format using XBee module – This was solved by digitising the integer and then sending it.
5. Sending the data in a string form over Java Sockets – This was solved by converting the strings to byte arrays and then sending them.
6. Power when provided directly at VBUS of the TM4C123GXL it would cause the robot to restart – The cause of the problem could not be determined. However the problem vanished automatically after two days.
7. Motor Driver IC would not run from the 5V coming from the Tiva board – The cause of the problem could not be determined. However the problem vanished automatically during testing.
8. The PIN PF0 was locked and we were unaware of this fact and tried using this port like any other pin – After a lot of research on the web we found how to unlock the pins.

Bibliography

yet to be done. :P

