



Trackless Tram System

Dr. Muhammed Elrabaa

Ayman Kurdi
Basil Hashim
Faisal Al-Kharboush

200849500
200830740
200825040

5/10/14

Senior Design
Project

Table of Contents

1. Introduction:	3
2. Problem Statement.....	4
2.1. Positive impact.....	4
2.2. Possible negative impact, due to misuse or unaccounted for risks.....	4
3. Project specifications	4
3.1. Requirements	4
3.2. Technical specifications:	5
4. Teamwork.....	5
5. Engineering Design:	7
5.1. Decision making	7
5.1.1. Steering mechanism:.....	7
5.1.2. Propulsion:.....	8
5.1.3. Steering (Motorized):	9
5.1.4. Steering (design & implementation):	10
5.1.5. Communication:	15
5.1.6. Decision conclusion.....	16
5.2. Architecture	16
5.2.1. Sub-function:	16
5.2.2. System architecture:.....	18
5.2.3. Components listing:.....	19
5.2.4. Component Functions & Component Interface:.....	20
5.2.5. Custom Components Design justification:	21
5.3. Implementation:.....	21
5.3.1. Setting:.....	21
5.3.2. Variables:.....	22
5.3.3. Commands and data sent to and between cars:	22
5.3.4. Reaction to received data:.....	23
5.4. System integration	23
6. Issues:.....	26

6.1.	Ordering the wrong parts	26
6.2.	Stepper motor controls.....	26
6.3.	Cutting the body of the car	27
6.4.	Creating a proper mechanical steering mechanism	27
6.5.	Utilizing the Fab-Lab:	28
7.	Engineering Tools and Standards:	28
8.	Conclusion:	28
9.	References:.....	30
10.	Appendix:.....	31

1. Introduction:

Most big cities use fossil fueled buses for public transportation for its residents. These buses produce large quantities of heat which contribute 18% of total emissions ⁽¹⁾. The emissions and the heat dramatically reduce the quality of life of both citizens and visitors of the town. However, it is not an easy task to switch to tram systems due to the unrealistically large initial capital investment required. The required initial investment can reach almost 500,000,000 dollars ⁽²⁾, so most of the initial required capital investment would go towards building the infrastructure. Although the electrical trams way solve buses issues, it is still infeasible to build a tramway system in crowded cities. The Trackless Tram System is an alternative public transportation system was proposed by Dr. Muhammad Elrabaa. In this report the team is going to describe the project as a new public transportation idea. Also, the team was working on this project during the semester to introduce a prototype.

The Trackless Tram System consists of a pilot car and an optional number of trailers cars. Unlike conventional trams, there is no limitation on the rotation angle of these joints since all the tram cars are steerable. Hence all cars can follow any path traveled by the pilot car even if it involves sharp turns. Also, all cars are equipped with normal bus-like rubber tires which means they can move up or down a hill, something conventional trams can't do. The pilot car could be driven by a human driver or a computer-based automated system. Each car is equipped with a power module, electric motors based propulsion system, a brake system, and front wheels steering system. Both the propulsion and brake systems are under the direct control of the pilot driver so all cars move, accelerate, decelerate and stop simultaneously and in exact synchrony.

¹⁾ ucsusa.com

²⁾ www.tahoetram.com

2. Problem Statement

This project describes a trackless (railway-free) multi-car electric tram (dubbed the tram bus) that attempts to solve the problem cost bringing it down to \$225M from \$500M and reducing emissions by moving away from the dependency on fossil fuel. This tram system should also be able to move freely inside the city.

Public transportation helps in reducing traffic, risk of accidents and emissions. However, having an efficient city wide transportation system that can go through traffic properly is hard to implement and costs tremendous amounts of money. This project aims to solve those problems.

2.1. Positive impact.

If successful, this project can provide large incentive to implement public transportation to cities that don't have one in place already. In addition to reducing traffic and emissions, it will also be relatively cheap to implement.

2.2. Possible negative impact, due to misuse or unaccounted for risks

Since this system should move freely within the city, it could cause unwanted traffic congestions if the driver moves around lanes too.

3. Project specifications

3.1. Requirements

According to proposed solution provided by Dr. Elrabaa', we are required to build a small concept of the original Trackless Tram System described in the introduction, following is the requirements that should be considered for the project:

- A Pilot car controlled by the user.
- Cars attached to the pilot car (trailers) and follow its path precisely, without any control by the user.
- Capable to do sharp turns.
- The pilot must share his speed and direction with his trailers.
- Extendibility of the number the attached cars.
- Make them start moving and stopping at the same time.
- The pilot must have a break system.

- The cars are equipped with normal bus-like rubber tires.
- The controllers of trailer cars must control the amount, position, and timing of steering of each car such that all cars follow in the exact path that was traveled by the pilot.
- There must be a fixed distance between the cars, so they would not crash with each other.

3.2. Technical specifications:

- The system consists of two types of cars pilot and trailer.
- The system provide a user interface to control pilot car.
- The system have one pilot and at least two trailers cars. The pilot is controllable by the user.
- Each car is responsible of sending distance they covered every 10ms or 50mm whichever comes first.
- Each car is responsible of sending the angel at which they are moving and at what point did they change to that angle. Angle should be precise to a 10°.
- All cars should be connected together at all times.
- Distance between cars should not be less than 50mm or more then 100mm.

4. Teamwork

Our team had a clear unity of purpose, the team has discussed the objectives until everyone committed himself with them. We divided the project into milestones, to clarify the objectives more and infesize the commitment, assigning each of which task to a member, who is the most experienced and capable of doing that task with ease, as a leader to this task. The rest of the team of course support him and work in the subtasks. The following table lists the tasks, subtasks, the task owner and the duration:

Task id	Task	Members Assigned	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1.0	Hardware	A		x	x	x	x										
1.1	Deciding	A,B,F		x													
1.2	Ordering	F			x	x											
1.3	Hardware setup	A					x										
2.0	Car Modeling	B				x	x	x	x	x	x						
2.1	Body Shape	F,B				x	x	x									
2.2	Steering Mechanism	B							x	x							
2.3	Fabricating the designed models	B							x	x	x						
3.0	Software	B				x	x	x	x	x	x	x	x				
3.1	Speed and angel control system	A,B,F						x	x	x	x	x	x				
3.2	Feedback system	B,F					x	x	x								
3.3	Communication system	F					x	x									
4.0	Communication setup	F					x	x									
5.0	Finalizing project	B										x	x	x	x	x	x
5.1	Testing	A,B										x	x	x	x		
5.2	Debugging	A,B,F											x	x	x		
5.3	Demo	A,B,F														x	x
6.0	Report	A,B		x	x	x	x	x	x	x	x	x	x	x	x	x	x
6.1	Documenting	A,B,F		x	x	x	x	x	x	x	x	x	x	x	x		
6.2	Presentation file	A,B,F							x	x					x	x	x

Table 1: Team Tasks Sqedual during the semester

We didn't have a leader for the project instead we had 3 leader each one is a leader for the phase he is responsible for, each phase has its leader, that actully enhance the working atmosphe and pushed the team to perform effectievly and informaly and work more effectivly. The door of criticism was open to whoever see something that needs to be changed and everybody was free to express his feelings and ideas. We beilive that we were performing as an effictive team.

5. Engineering Design:

5.1. Decision making

In this section we discuss the major design decisions that we have made and we compare them to the other solutions we considered and why we chose a particular solution over another.

5.1.1. Steering mechanism:

Here we considered two possible choices: Differential drive system or add a motor for steering.

Criteria: mechanism most representative of the real world.

5.1.1.1. Examined options:

5.1.1.1.1. Differential Drive:

We considered differential drives for steering because they are cheaper to deploy and implement but the problem with it is it allows for very unnatural turns compared to existing transportation methods.

5.1.1.1.2. Additional motor for steering:

Having an additional motor whose sole purpose is steering would definitely increase the cost per car. However, doing so would allow for greater control over steering and will allow us to ultimately have two separate systems for steering and propulsion. Employing this mechanism would also be more familiar to steer and have a parallel in the real world in everyday cars.

5.1.1.2. Tradeoffs:

Here is a table to show the factors we considered to choose one of these two mechanisms:

	Differential Drive	Additional Motor
Driver Familiarity	Can be awkward	Similar to real cars
Tightness of controls	Can Turn in place	Very tight controls
Price*	No additional parts	Needs additional parts
Final Decision		Additional Motor

Table 2: Comparison between two steering methods

5.1.1.3. Final decision

Having an additional motor was the more attractive choice for this particular implementation. Namely because it allowed for separation of the two systems (propulsion and steering) and for the added benefit of representing the real world more accurately than differential drives.

5.1.2. Propulsion:

While considered a choice for propulsion we had one major requirement: speed feedback. The need for speed feedback stemmed from the fact that we had to synchronize the speed of the two trailing cars with the leading car.

Criteria: considerable speed, speed feedback is a major plus.

5.1.2.1. Examined options:

5.1.2.1.1. Stepper motor:

More expensive than its DC counterpart and slower rotational speed. However, a stepper motor allows us to control the number of rotations we wanted, Thusly giving us a sizable amount of control over speed making synchronization easier to implement. One of the problems with stepper motors however is we don't actually have feedback. If a motor is slower than another we would have no way of knowing without implementing extra sensors. Another concern was bumps, you could tell a motor to turn a number of times but if it had something preventing it from turning (bumps) the motor would still consider the job done even though it was prevented from doing so.

5.1.2.1.2. DC Motors:

DC Motors have no natural way of controlling speed precisely implemented in them. Though Pulse Width Modulation (PWM) can give some control over speed. Speed feedback is also not present in DC motors. However, we found ready solutions that give feedback on how many rotations a tire made implemented around DC motors.

5.1.2.2. Tradeoffs:

Here is a table to show the factors we considered to select one of these DC motors for the back wheels:

	Feedback wheel encoder set + dc motor	DC motor GM14a	DC motor GM12a	Stepper motor
Speed		Fast (100rpm)	Fast (170 rpm)	fast
Price	39\$+ dc motor\$	15\$	16\$	16\$
Accuracy (feedback)	very high	-	-	low
Power consumption	moderate	moderate	moderate	high
Torque		30oz	10oz	
Final decision	Feedback wheel encoder set + dc motor			

Table 3: Comparison between different DC motors

5.1.2.3. Final Decision:

Keeping in mind that a stepper motor can turn friction (or lack thereof) can cause differences in speed between the different cars. Seeing as both motors have no real way of providing feedback naturally, we decided to go with the DC motor because ready-made solutions being available. The fact that it was smaller in size was a plus.

5.1.3. Steering (Motorized):

Control over the angle of steering a major concern. Implementing an additional tire and motor was a step towards achieving that. However the kind of motor we implemented here was also an important factor.

Criteria: turn degree accuracy.

5.1.3.1. Examined options:

5.1.3.1.1. Stepper Motor:

Stepper motors can be controlled by moving it a number of steps (n) out of the total number of steps for a rotation (N) which allows for various degrees of control n/N .

5.1.3.1.2. DC Motor:

Using a dc motor would allow us to have 3 different steering directions only: left, right and straight ahead. These are not ideal in representing real life scenarios.

5.1.3.2. Tradeoffs:

Here is a table to show the factors we considered to select either a DC motor or a stepper for the steering:

	DC motor	Stepper motor (200 step motor)
Accurate rotation	continuous rotation	high (1.7 degrees)
Real life representation	low representativeness	high representativeness
Final decision		Stepper motor

Table 4: Comparison between DC motor & stepper motor

5.1.3.3. Final Decision:

Using a stepper motor was the more obvious choice here given the requirement of representing real life scenarios since it represents a wide range of angles rather than just three.

5.1.4. Steering (design & implementation):

Some problems were faced in the design phase and other in building and manufacturing the design we are going to discuss them in the issues. To begin with the:

5.1.4.1. Steering Design:

Immediately after we got the car body, we started to think about steering using a single wheel because with that body, showing in Figure-3. It'll be easier to build and also requires less resources as it'll be the same mechanism as the two wheels. We come up with different ideas of single wheel steering using stepper motors. We discussed them with some excellent, 1st honor, mechanical engineering students from both graduate and under graduate. Finally, we go a design, shown in Figures 1&2, based on Ackerman steering geometry ⁽³⁾, but we just modulated it for a single wheel.

⁽³⁾ Is a geometric arrangement of linkages in the steering of a car or other vehicle designed to solve the problem of wheels on the inside and outside of a turn needing to trace out circles of different radius [Wikipedia].



Figure 1: The car's body we bought

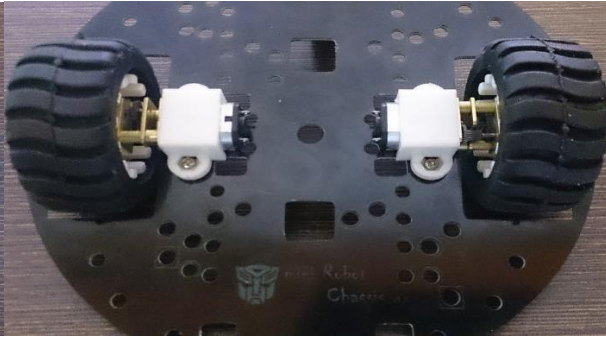


Figure 2: The body after installing the back wheels

The stepper motor has a weak torque which is not enough to hold and steer a wheel without gears. So, with the help of mechanical engineering students we came up with rack and pinion idea.

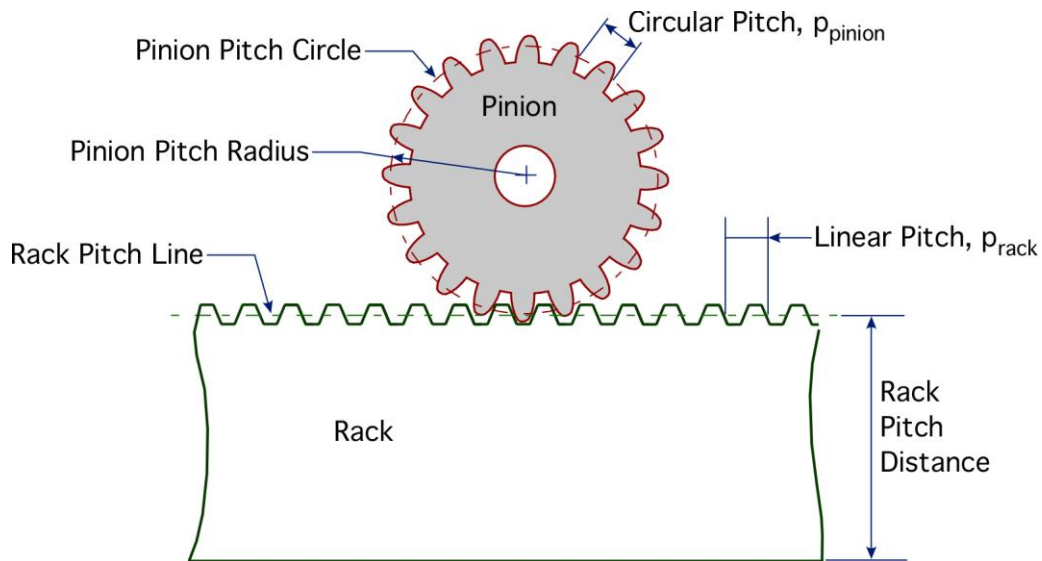


Figure 3: The Rack & Pinion principle

In the design we used the Inkscape software, because it's an open source software. The design is as follows:

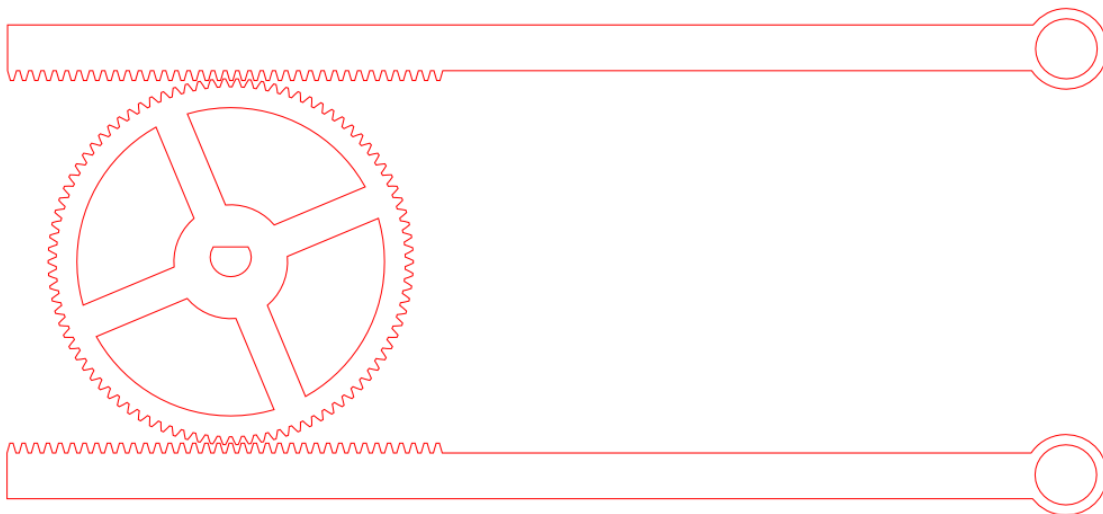
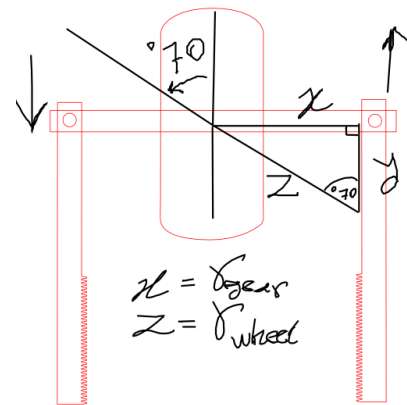


Figure 4: Inkscape design for the Rack & Pinion in 2D

The measurements of the design are based on simple trigonometric equations described as follows:

As the maximum degree of turning right/left is 70° we need to determine the shaft length ($2x$) which also give the diameter of the pinion gear and the also the rack. This all could be determined by:

$$x = z \sin 70^\circ \quad ; \text{ given } z$$

- Where x half the shaft length.
- And z is the wheel radius.

We have z as 34 mm ; because of car height.

$$x = 34 \sin 70^\circ = 15.98 \approx 16 \text{ mm}$$

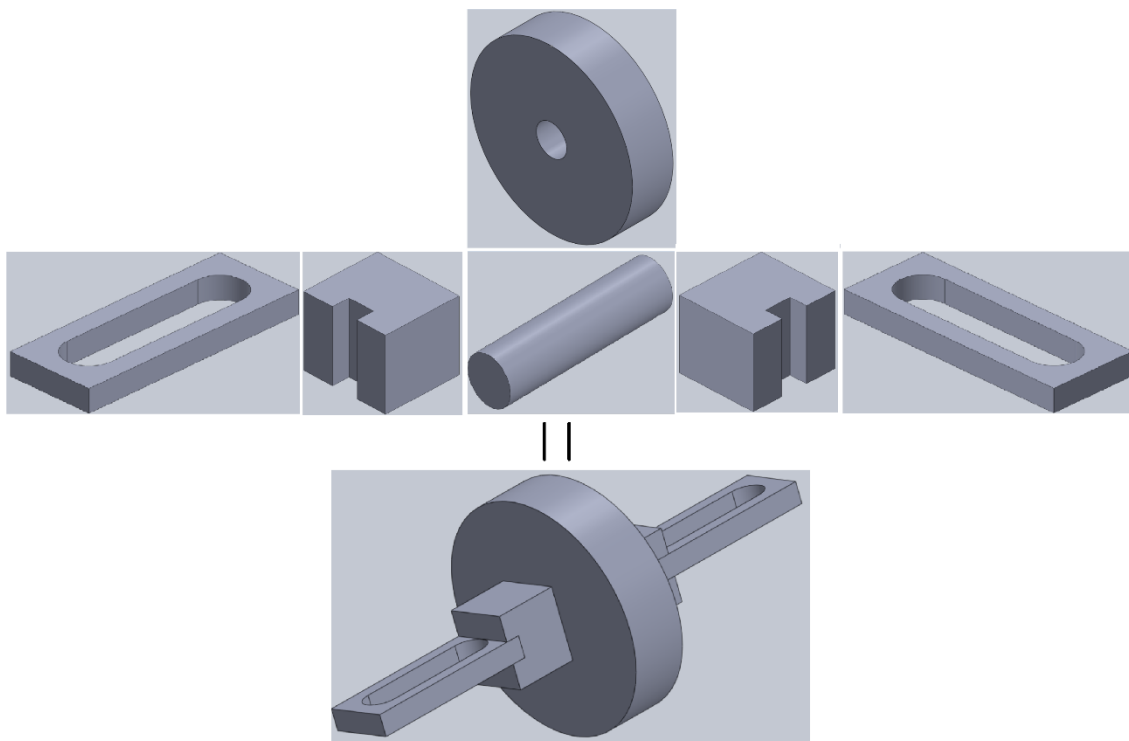
The length of the shaft diameters the pinions diameter:

$$2x = 32 \text{ mm}$$

The length of rack doesn't depend on anything [Figure 5: Description of the calculation](#) but the material of the rack itself.

The rack and pinion gears design is ready to be manufactured in the Fab-Lab.

At that point, we designed the front wheel and the shaft but the design haven't been built on a 3D software yet, and none of us have experienced working with 3D design. With the help of mechanical engineering student we build our design in the SolidWorks. The result of the design is as follows:



[Figure 6: The parts we designed and assembled together to get the front wheel](#)

Now the whole designed is ready to be manufactured.

5.1.4.2. Steering Implementation:

We booked both the 3D printing and laser cutter machines in the Fab-Lab. The result of the Fab-Lab is the following:

- The laser cutter:

We got clean parts of the rack & pinion gears ready to be used.

- The 3D printer:

Actually, we needed to clean the parts we got out of the 3D printer.



Figure 7: The 3D printed parts before and after we cleaned them

The entire system as connected together is as follows:

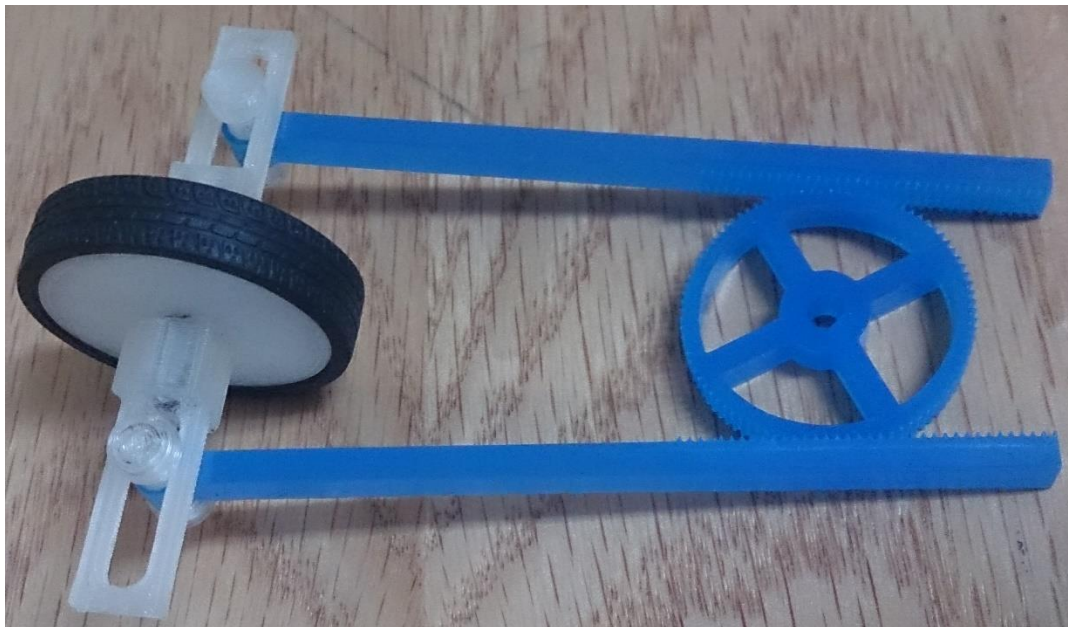


Figure 8: Steering system without the motor

5.1.5. Communication:

Communication between the tram system vehicles is key in synchronizing their movements. The requirements that dictated the decisions was cost and time propagation delay. Built in synchronization would be a plus too.

Criteria: does not limit the cars' movement (wireless) and covers the distance between the 3 tram cars (3M radius).support mesh network.

5.1.5.1. Choices considered:

5.1.5.1.1. ZigBee

ZigBee is a specification for a suite of high level communication protocols used to create personal area networks built from small, low-power digital radios. ZigBee is based on an IEEE 802.15 standard. Though low-powered, ZigBee devices can transmit data over long distances by passing data through intermediate devices to reach more distant ones, creating a mesh network; i.e., a network with no centralized control or high-power transmitter/receiver able to reach all of the networked devices. The decentralized nature of such wireless ad hoc networks make them suitable for applications where a central node can't be relied upon

5.1.5.1.2. WiFi:

Can connect to a network resource such as the Internet via a wireless network access point. Such an access point (or hotspot) has a range of about 20 meters (66 feet) indoors and a greater range outdoors. Hotspot coverage can comprise an area as small as a single room with walls that block radio waves, or as large as many square kilometers achieved by using multiple overlapping access points.

5.1.5.1.3. Bluetooth:

Bluetooth is eliminated because it can't support mesh network which is required as stated in the technical requirement.

5.1.5.2. Trade offs

Here is a table to show the factors we considered to select either WiFi or ZigBee:

	WiFi	ZigBee
Mesh	yes	yes
Coverage	20 – 100 M	10-20 M
Price	40\$	19\$
Power consumption		Relatively lower
Final decision		ZigBee

Table 5: Comparison between WiFi & ZigBee

5.1.5.3. Final Decision:

Both choices provide a mesh network solution. Although WiFi looks superior in coverage radius, ZigBee meets our needs for building the prototype plenty. Not to mention it is much cheaper.

5.1.6. Decision conclusion

All equipment were selected specifically to meet the technical requirement of this project. Arduino UNO provides an easy programmable interface. XBee can provide mesh networks and packet or serial communication. DC motor are small light have high speeds and less power consumption and is ideal for speed control. Stepper motors can turn in accurate angels and is ideal for steering the car.

5.2. Architecture

5.2.1. Sub-function:

5.2.1.1. Control:

Our control system is made up includes:

- Laptop (keyboard for input)
- Micro controller unit
- ZigBee

This system is dedicated to control the pilot car using simple inputs from the arrow keys on the keyboard. These commands include:

- Forward propulsion speed.
- Turn angle.
- Break.

This subsystem is connected to the propulsion subsystem and the steering subsystem on the pilot car only.

5.2.1.2. Propulsion system:

This subsystem consists of:

- 2 DC motors per car.
- 2 drivers with light sensors.
- 2 geared tires.
- Arduino UNO to derive PWM signal.

The propulsion subsystem controls the speed of the car while also providing feedback using the light sensors regarding the speed of the car. This subsystem is connected to: synchronization subsystem.

5.2.1.3. Steering:

This subsystem consists of:

- Stepper motor (200 steps).
- Custom made single wheel control mechanism.
- Arduino UNO to derive step pulses (angles).

The purpose of this subsystem is to control the turn angle of the car as precisely as possible. This system can control the degree to a .33 of degree

This system is connected to synchronization subsystem and the control subsystem on the pilot car, however, it is only connected to the synchronization subsystem on the follower cars.

5.2.1.4. Communication:

This subsystem consists of:

- laptop
- Arduino UNO
- ZigBee explorer board (USB)
- ZigBee wire antenna
- ZigBee shield

This subsystem is what connects everything to everything. Although it serves a basic function, it allows us to have nonrestrictive wireless communication where we need it.

Communication subsystem is connected to: Control sys. Propulsion and steering (pilot car only), and synchronization system.

5.2.1.5. Synchronization:

This subsystem consists of:

- Feedback from the DC drivers.
- Arduino UNO

Synchronization by in each car separately (distributed). The pilot car provides the needed information to the follower cars and they then synch their speed and angle accordingly after performing the synchronization function within each of the cars.

5.2.2. System architecture:

- Each car has three wheels two in the back, one in the front.
- All cars are equipped with an Arduino UNO microcontroller responsible for controlling components of that car.
- Each car is equipped with two DC motors responsible for controlling the speed of the car. The power to those motors is controlled by the microcontroller to define their rotation speed.
- Each car is equipped with a stepper motor that is responsible for the steering angel and is controlled by the microcontroller.
- Each DC motor is equipped with a light sensor to read how many rotation the wheel finished, this is done by counting the white and black areas.
- Each car is equipped with an XBee module connected to the microcontroller to take care of the communication between the cars.
- A custom steering mechanism is designed using the Fab-Lab and attached it to the body of car.

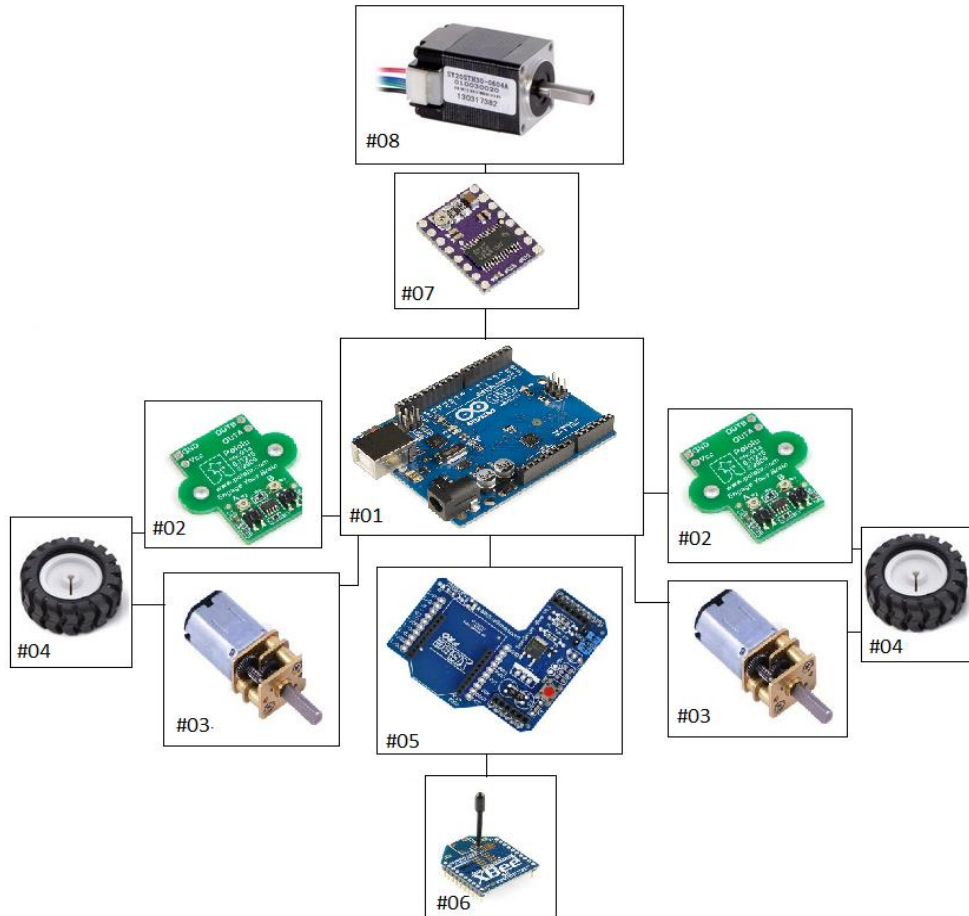


Figure 9: Car's Architecture

- An XBee connected to a computer to support user input to the pilot car.
- A terminal in the Computer is used to input speed and angel.

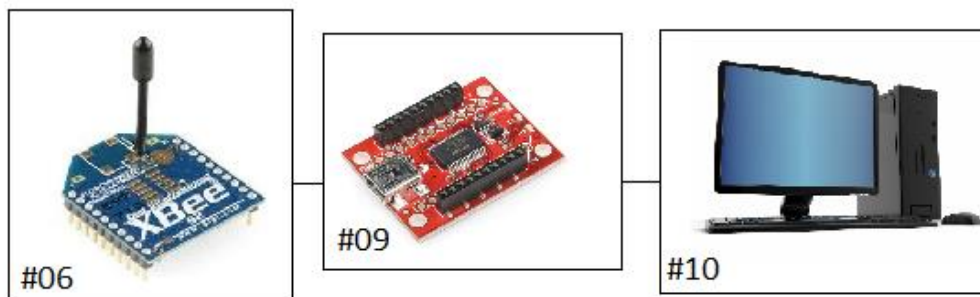


Figure 10: Component used for communication

5.2.3. Components listing:

Components from the figure:

#01 Arduino UNO

- #02 Encoder for Pololu Wheel
- #03 50:1 Micro Metal Gearmotor HP
- #04 Pololu Wheel 42x19mm
- #05 XBee Shield For Arduino
- #06 XBee 1mW Series2
- #07 Stepper Motor Driver Carrier, Low Current
- #08 Stepper Motor: Bipolar, 200 Steps/Rev, 20×30mm, 3.9V, 0.6 A/Phase
- #09 XBee Explorer USB
- #10 A computer

Other components:

- #11 Micro Metal Gearmotor Bracket Extension
- #12 Custom wheel
- #13 Custom Steering Gear
- #14 Custom Steering racks
- #15 Custom Steering shaft
- #16 Custom Steering pinion
- #17 Custom Car body
- #18 Battery holder + batteries
- #19 Breadboard

Software components:

- #20 X-CTU
- #21 Arduino IDE
- #22 Inkscape
- #23 SolidWorks
- #24 Belnder 3D

5.2.4. Component Functions & Component Interface:

“Using the same numbering from previous section”

#01 multipurpose microcontroller “for more on purpose read implementation section”.

#02 Used to read feedback from the wheels.

#03 Responsible for car speed.

#04 Used as back wheel.

#05 Needed to interface between XBee & Arduino.

#06 Used to for communication uses IEEE 802.15 standard as communication interface.

#07 Used to interface between Arduino and the stepper motor.
#08 Responsible for car movement direction.
#09 Used to interface XBee with pc.
#10 Used to control pilot car.
#11 Used to hold the back wheels and the Encoders to the car body.
#12 Used as front wheel.
#13 to #16 Part of the steering module used to control movement direction.
#17 Used to hold car parts together.
#18 Needed to provide power to the system.
#19 Connects parts together
#20 Used to configure XBee, also used to send commands to pilot car.
#21 Used to build and upload sketches to the Arduino.
#22 Used to design the racks and pinions. In format ready for the laser cutter in the FABLAB.
#23 Used to design the front wheel and shafts.
#24 Used to add support for the files generated from SolidWorks and make it ready for the 3D printer.

5.2.5. Custom Components Design justification:

The only part we used custom component is the car's steering mechanism. We needed to use customized component for the steering mechanism because there was no existence of something that fits a toy and operate like a car. The components are:

- Rack & Pinion gears.
- Front wheel with its shaft.

Note any components not listed here is an off-the-shelf component:

5.3. Implementation:

5.3.1. Setting:

- After assembling all car parts together parts together.
- Distance between cars is determined by $(21 * \pi^2) / 12 \text{ mm} = 17.27 \text{ mm}$
We will call this as unit meter um ". This is the accuracy at which the encoder can read distance.
- Cars at the beginning will be aligned and have a distance between them equal to 8 um .

- Network is set to AT mode. Settings as follows:









PC	Pilot Car	Trailer Car 1	Trailer Car 2
Router PAN ID: 485 DL:0x0010	router PAN ID: 485 SL:0x0010 DL:0x0020	router PAN ID: 485 SL:0x0020 DL:0x0030	Coordinator PAN ID: 485 SL:0x0030
			
			

Table 6: With this settings each car can communicate with the next car as required

5.3.2. Variables:

- A Variable for the angel is initialized to 0 at all cars.
- Only pilot car has a speed integer variable and is initialized to 0.
- An integer variable for distance covered should be initialized as follows:

Pilot Car	Trailer Car 1	Trailer Car 2
16	8	0

Table 7: Initialization of integer variables

- **Note: This variable will be incremented based on readings from the encoder**
 - The trailer cars will have a 2D array queue that will list the next angel change and the point at which the change will happen.

5.3.3. Commands and data sent to and between cars:

- PC can chose between stopping and 5 different speeds by sending a numeric character through the terminal '0'=0v '1'=3v '2'=3.5v '3'=4v '4'=4.5v '5'=5v.

- PC can chose between different angels by sending a character as follows:
 - Left: ['q' 30°, 'w' 20°, 'e' 10°].
 - Direct: ['r'].
 - Right: ['t' 10°, 'y' 20°, 'u' 30°].
- Each car will send an integer to the next car.to update the distance covered.
- Each car will send a char to the next car followed by an int.to tell of next angel change.
- Data transmission happens every 10ms or whenever is invoked.

5.3.4. Reaction to received data:

Pilot:

- Pilot will act immediately by changing speed or angel. And then wait for 5ms.

Trailer:

- An integer is received: Trailer will compare distance he covered with distance preceding car covered and set speed in an appropriately. Trailer will also check with the queue to see if making a turn is needed. And then wait 5ms for next read.
- A char is received: Trailer will read next integer and append data to end of queue. And then wait for 5ms for next read.

5.3.5. Emergent invocations:

- When the difference in covered distance is less than 4um.Stop immediately. And invoke data transmission.
- When the difference in covered distance is more than 12um.speed up to max speed. And invoke data transmission.

5.4. System integration

Testing:

Communication:

- **Test1:** To check for successful communication a sketch was uploaded to each component of the network where the shout a

specific string periodically. Successful communication is achieved when all network components are able to communicate as desired.

- **Test2:** Another test is to use the XCTU search for connected nodes function.

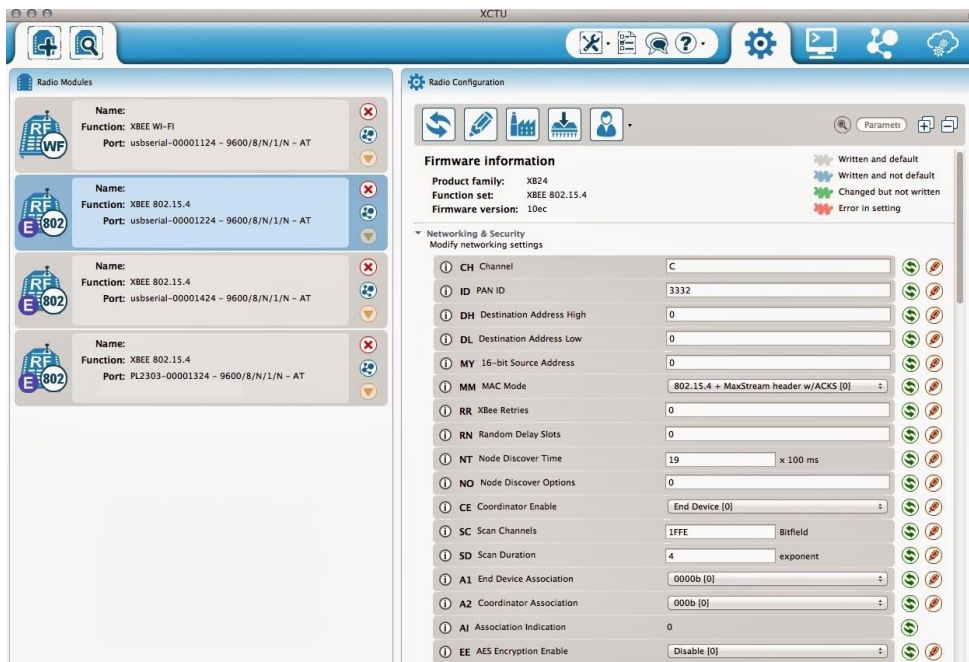


Figure 11: Screenshot of XCTU software

Successful communication is achieved since XCTU is able to find all components as shown in figure

Feedback system:

Test1: we will need to make sure that the DC motor are identical to make sure they won't cause differential drift.

Unit to measure distance is $(21 * \pi^2) / 12 \text{ mm}^{(4)}$

The result	Test subject1	Test subject2
5v	104,103,104	103,103,103
4v	83,83,83	83,82,84
3v	62,62,63	62,62,64

Table 8: Feedback results

⁽⁴⁾ This is the distance at which the encoders give as feedback

- Note: this test could be slightly affected by wrong positioning of wheel at beginning of reading.

Using three takes per motor with different voltages it seems they are fairly identical. The differential drift at worst case will be 17.27mm for every 100m.

Debugging:

Stepper motors and stepper driver:

We had a problem running the stepper motor using the drivers we had, so we did debugging for the motors and the drivers separately:

Motors:

We removed the motors from the drivers and applied direct voltage (5v) on them. This has caused the three motors to stutter and turn with a little help. The problem was that our controller was not driving enough current: tested current 30 mA while required current was 600 mA.

Drivers:

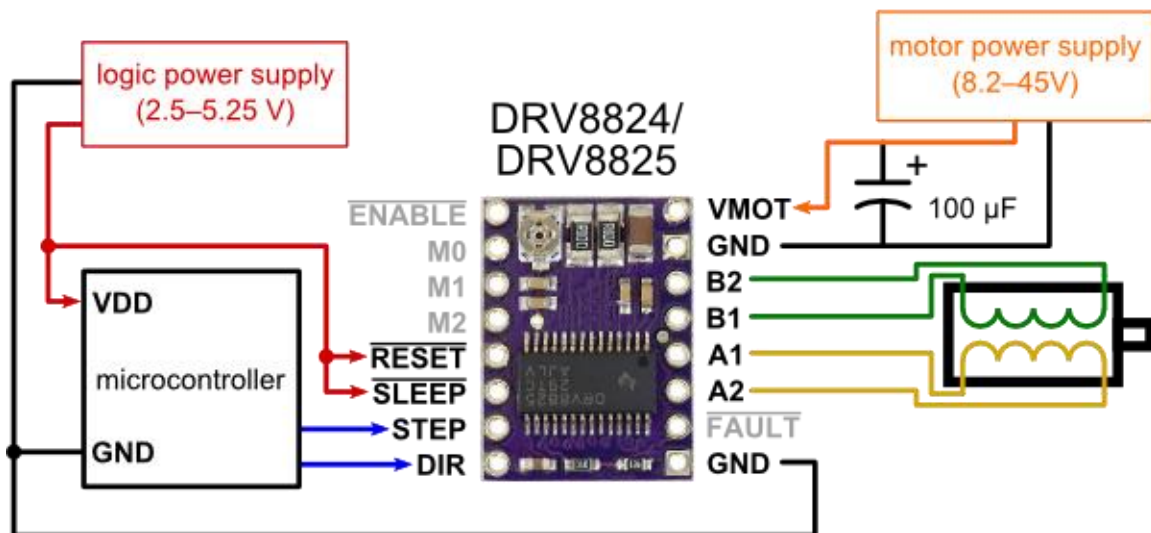


Figure 12: Driver schematic

We set up a testing grounds for the drivers and continued to all pins on all three drivers. We found out that one of the output pins of one of the drivers were busted and was not giving out any output.

Each pulse to the step pin should give us one step from the motor, however the motor would not move in correspondence with the signal we were sending to the step pin.

Fault should drive high if we test it if the driver was connected properly.

We tried sending a continuous digital signal through the step pin using Arduino's `analogWrite (pin, pulseWidth)` where `pulseWidth` is the active duty cycle time. This signal should have resulted in a continuous rotation in the stepper motor. However there was no rotation whatsoever. We should also mention that the fault signal was driving high.

Another thing we tested was to send a single pulse to the step pin every set amount of time⁵.

we changed the time from 50 uS to 1 mS. However the motors still wouldn't move properly, fault was driving high (driver chip is working A-Okay) and the pins were output pins to the motor were driving proper voltage. However the motors would still not work.

6. Issues:

This section is dedicated to discussing major setbacks and problems faced in general during the implementation of the prototype.

6.1. Ordering the wrong parts

Problem: at first we bought only two cars parts. And when were sure those are the parts we will use we ordered the third car parts. The issue is that the two DC motors we got were different.

Issue discussion: this was not a big issue because they still have the same size and shape they also have same voltage input and almost produce equal result speed and torque.

Final solution: since the speed of the pilot car is managed in a different manner as shown in the implementation section. We used those two motors in the pilot car.

6.2. Stepper motor controls

Problem: motor would not run.

⁵ See the code in the Appendix

Attempted solutions: we tried changing the motor and changing the driver before we went into more in-depth debugging to find out the root cause of the problem. Check debugging section for more information.

Final solution: unfortunately having realized one of the driver chips was not working properly which made us realize that we should ignore this problem for the time being.

6.3. Cutting the body of the car

Problem: We needed to modify the car's bodies and cut a hole through them for the stepper motor. The cars have to be identical in the cutting in order to synchronize the movement with ease. The body of the car was made of carbon fiber which made laser cutting it practically impossible.

Attempted solution: unfortunately, we attempted laser-cutting the body thinking that the laser cutter will be able to make it. This had caused the body to deform in a way that made us unsure of the weight distribution.

Final solution: Eventually, we redesign that that part on the same pattern and made adjustments to account for the weight difference in the final prototype. Then, we build our own bodies, this time from acrylic, because it's strong enough and quite light.

6.4. Creating a proper mechanical steering mechanism

Problem: there was no available single tire or double tires off-the-shelf that would fit the size we were using for our prototype.

Attempted solution: even though we did not prefer it, we considered going back to using differential drive, but in the decision making phase, the team decided to make the car acts more like a real car rather than a toy car, because we want to implement this project as a real life prototype and not for the sake of the project. Therefore, the assigned member needed to meet with mechanical engineering student to come up with something fits a toy and operate like a car. After five or more meeting he came up with Ackermann steering mechanism, described above.

Final solution: we ended up creating a dramatic looking single gear system that would allow us to use a stepper motor to control a single tire in the front of the car.

6.5. Utilizing the Fab-Lab:

Problems: there are actually some problems we face utilizing the Fab-Lab. Many times we implement the design and it doesn't print it properly. I may should say many other times we just can't use a machine in a certain zone because the guy in that zone is having lunch during his working hours or absent and nobody is there to cover for him, and you can't do the job by yourself even if you've taken the workshop. Also, the Fab-Lab doesn't open for men in the weekends and Sundays until 4pm, although that's not mentioned in their website. So, if we progressed in the design in a weekend we'll have to wait until Sunday 4pm. Summarizing this problem, in spite of its usefulness, it waists a lot of time.

7. Engineering Tools and Standards

7.1. Engineering Tools:

7.1.1. Debugging Tools:

Variable Power Source

Oscilloscope

Millimeter

XCTU

7.1.2. Design Tools:

SolidWorks

Inkscape

Arduino IDE

7.2. Standards:

ZigBee: IEEE 802.15

USB 2.0

8. Conclusion:

This paper show a feasible solution to the increasing problem of public transportation by implementing a prototype module of a wireless trackless tram system. We believe that this solution can be implemented in different cities for a tremendously cheaper price than the currently available solutions. We hope to see this design concept adopted in the real world.

The project had presented a prototype for the Trackless Tram System. During the semester we faced different problem starting by collecting the equipment's

up to taking serious decisions. The team was working hard to finish the system module. We admit there was a tremendous amount of stress and work during the process of completing it, but we are so glad we finished a working module of the system.

In addition, we learned many thing in different fields especially in Computer and Mechanical Engineering fields. We also became experts in general engineering techniques how important of reading and research before taking decisions.

9. References:

- 1- Clean vehicles, retrieved on 13 march, 2014 from:
http://www.ucsusa.org/clean_vehicles/why-clean-cars/global-warming/
- 2- Costs and Revenues, retrieved on 13 march, 2014 from:
<http://www.tahoetram.com/costs-revenues>
- 3- From Wikipedia, the free encyclopedia
http://en.wikipedia.org/wiki/Ackermann_steering_geometry
- 4- The Senior Project Proposal.


```

{
xbee.readPacket(); //Reads all available serial bytes until a packet is parsed, an error occurs, or
the buffer is empty.

if (xbee.getResponse().isAvailable()) { //something available?

    if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE) { //is it rx packet?

        // now get the payload
        xbee.getResponse().getZBRxResponse(rx);

        Serial.print("Receive packet from (64-bits address) -> "); //type: XBeeAddress64
        Serial.print(rx.getRemoteAddress64().getMsb(), HEX);
        Serial.print(":");
        Serial.println(rx.getRemoteAddress64().getLsb(), HEX);

        Serial.print("Packet length is -> ");
        Serial.println(rx.getDataLength(), HEX); //getDataLength() type: uint8_t, Returns the length
of the payload.

        //rx.getData();
        //print the payload
        Serial.println("The packet ...");
        Serial.print("<< ");
        for (byte i=0;i<rx.getDataLength();i++)
            Serial.print(char(rx.getData(i)));
        Serial.println(" >>");
    }
}
}
}

```

```

/*
This is Xbee Transmitter
*/

#include <XBee.h> //include Header File
#include <Time.h>

#define MSG_LEN 18 //Message Length
XBee xbee = XBee();// Make a Xbee Variable
uint8_t payload[MSG_LEN];

//XBeeAddress64 addr64 = XBeeAddress64(0x0013A200,0x409022C7); //define 64-Bit Xbee
Adress of the Remote Host

XBeeAddress64 addr64 = XBeeAddress64(0x00000000,0x0000FFFF);

//XBeeAddress64 addr64 = XBeeAddress64(0x0013A200,0x408CC72B);

ZBTxRequest zbTx = ZBTxRequest(addr64, payload, sizeof(payload)); //Make a Xbee Packet

int led = 13;

void setup()
{
  xbee.setSerial(Serial);
  Serial.begin(9600);
  //pinMode(led, OUTPUT);
  Serial.println("Simple Xbee Communication - Transmitter");
  setTime(12,10,00,4,3,2013);
  char tmp[] = "COE DEPARTMENT";
  for (byte i=0;i<13;i++)
    payload[i]=tmp[i];

```

```
}
```

```
void loop()
```

```
{
```

```
  xbee.send(zbTx);
```

```
  //flash led
```

```
  // digitalWrite(led, HIGH);
```

```
  delay(1000);
```

```
  // digitalWrite(led, LOW);
```

```
  // delay(3000);
```

```
}
```