Smart Traffic Prediction System

-Discovering the wireless network -

Final Project Report

Group 3

TU Berlin
Ad-Hoc and Sensor Networks Lab

Telecommunication systems
TU Berlin
https://www.tu-berlin.de

Title:
Smart Traffic Prediction System

Theme:
Wireless Networks

Project Period:
Winter Semester 2016-17

Project Group:
Group 3

Participant(s):
Anirudh Shetty
Sarjo Das
Yatindra Shashi

Supervisor(s):
Prf. Vlado Handziski
Moksha Birk

Copies: 1

Page Numbers: 18

Date of Completion:
March 2, 2017

Abstract:

A prediction based traffic management system, which not only reduces traffic further ahead in the route taken, but also gives live traffic updates and alerts.

Contents

Preface

This document is the report concerning the project done in winter semester 2016-17 by our group. The group was composed of three international students who studied Ad-hoc and Sensor Networks on the 3 semester during our Masters at TU Berlin. The project was completed with advises from our professor Prof. Vlado Handziski and help from Moksha Birk.

TU Berlin, March 2, 2017

Anirudh Shetty (0387650)
<shetty@campus.tu-berlin.de>

Sarjo Das (387600)
<sarjo.das@campus.tu-berlin.de>

Yatindra Shashi (386845)
<y.shashi@campus.tu-berlin.de>

Chapter 1

Introduction

For our project we have developed a prediction based traffic management system, it mainly consists of traffic controllers, a unit that is sensible to traffic conditions on different parts of a interested area and adjust the traffic on next junctions accordingly to minimize future traffic congestion.

In our project we try to simulate the traffic mainly on those roads which are going to crossover at some intersection, and with the use of either sensors or other radio communications, we adjust the traffic on the roads based on both the traffic on current road in question and traffic in nearby places. We decided to have a threshold level for the traffic condition, above which we consider the road will be congested with traffic. Using this threshold and traffic count, we are able to generate the appropriate signals to divert and manage the traffic on relevant roads.

The software consists of three parts:

1. Central Traffic Management: Gives live feed of traffic in all parts, and also lets us know which areas are being managed and volume of current traffic being managed. And this unit also lets users to set and check traffic threshold for a whole area, by setting it on any one of the traffic controllers in the concerned area.

2. Traffic Controllers (TC): This is the system which polls and communicates with vehicles, other TCs and also Central Traffic Management system.

3. Vehicles: This unit replies to TC's polls to register itself under a TC, and get alerts from TC's about traffic, in scenarios with high probability of traffic, further ahead in it's route and current direction of movement.

The hardware consisted of three parts as well:

1. The Model: Where we simulate and test the traffic Management system.

2. Telosb Motes: This is combination of hardware, such as sensors and Transceivers, which we use in different combinations to either implement a vehicle, or a traffic controller or the main traffic management system.

3. A Laptop Terminal: Used to do serial snooping, to read the radio communications between all the motes, and represent it as a live traffic stats using python tools.

Chapter 2

High Level Design

Rationale: Whenever we go out now a days, we probably drive our vehicles, and if that is the case, there is a good chance we would have to deal with traffic congestion at some point of our travel. In coming years, this problem would only become worse, we believe big cities around the world needs a smart traffic detection system to reduce traffic congestion.

Also coming to current manual traffic management systems, it is known that huge number of road accidents take place each year due to human error, therefore implementing an automated system may help reduce human errors. Currently various developed and developing countries around the world are building smart cities, and smart transport system is an integral part of such smart cities.

So we came up with the idea, designing a smart traffic management system based on traffic prediction.

Logical Structure: The structure is divided in to manageable modules, and each module has multiple import functions. The traffic controller starts with preset threshold value and it will later manage traffic based on new values from central traffic management system and traffic near different traffic controller units, found using polling.

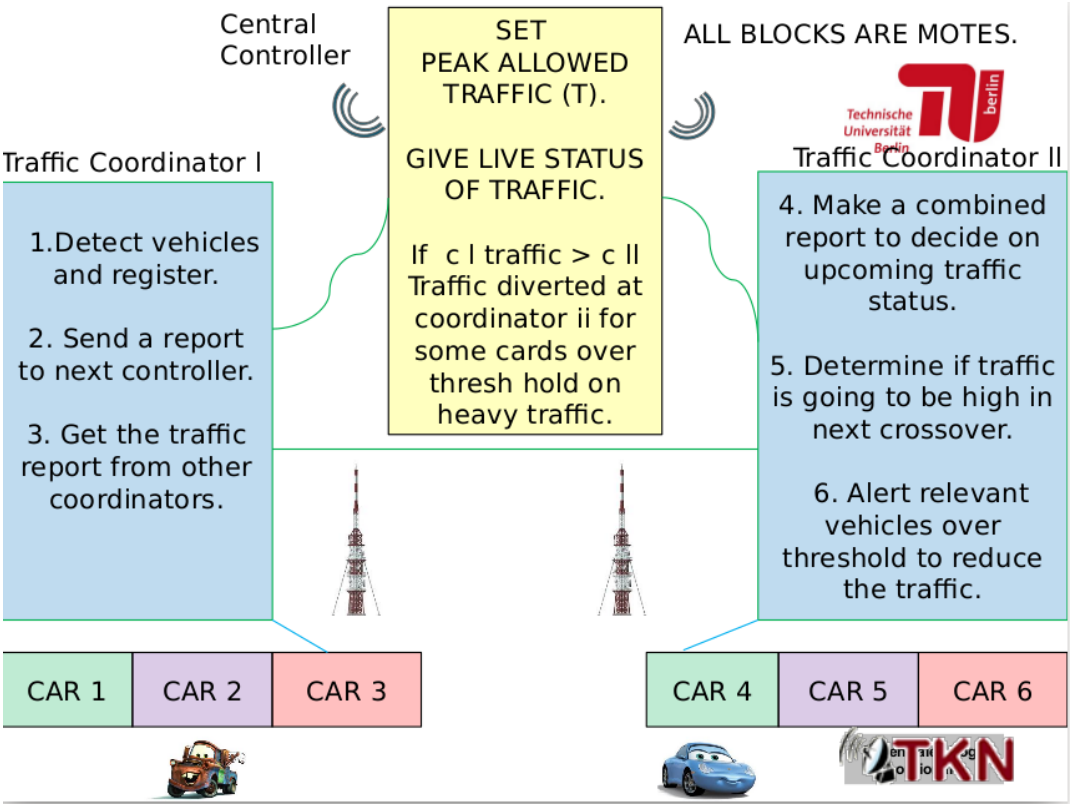

Figure 2.1: High level logical structure

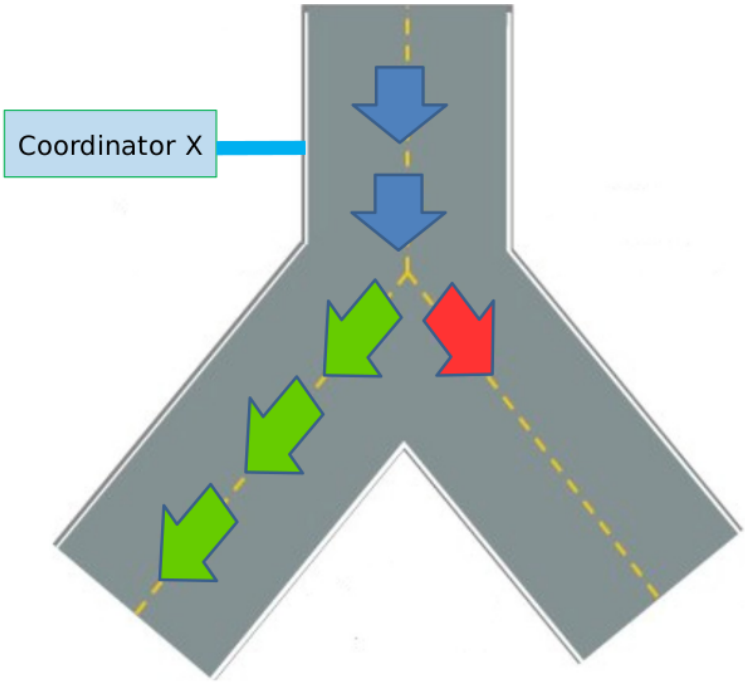

Figure 2.2: Vehicles being diverted

Hardware/Software Trade-offs: We have used both hardware and software in this project to accomplish our implementation goals. To determine the traffic density we used radio communications to poll the vehicles nearby, and by restricting the power of transmission in vehicles, we made sure the vehicles are counted only by one of the nearest traffic controller. This was combination of hardware and software parts.

For user input in central traffic management system, we use a UDP shell to communicate with a traffic controller, and propagate any changes in setting via multi cast to other traffic controllers. The vehicles basically respond to queries or alerts from traffic controllers via implemented software logic module.

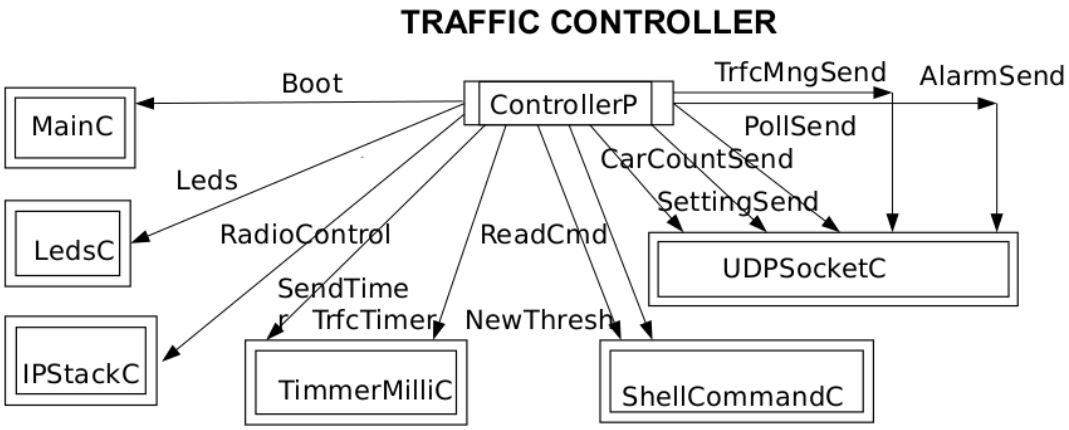

Figure 2.3: Traffic Controller Architecture

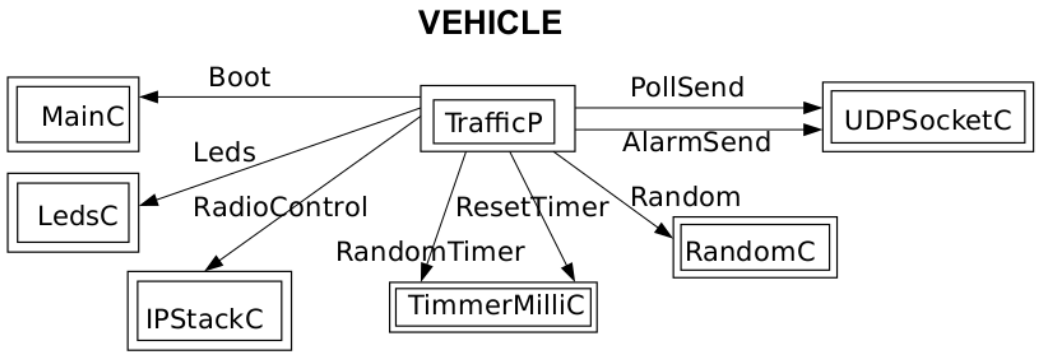

Figure 2.4: Vehicle Architecture

Chapter 3

Hardware/Program Design

Program Details: The most important and tricky part of the code is that of traffic controller. We had to detect a different vehicle count from each direction. At the same time, the intersecting and parallel roads in the same area are related to each other. Thus it's very hard to determine which vehicle comes under which traffic controller and how to convey the polling, traffic report and alert messages between the traffic controllers and traffic management system and also between the vehicles themselves, and it's tricky in terms of how to do the math and determine the traffic condition and also determine the vehicles which gets diverted.

In our design we separate all the asynchronous inputs from the normal traffic management system. There are two pieces which are just added on to the regular system and thus doesn't change its functionality by much. The first part is the user inputs, we utilize the UDP shell on the central traffic management system to define the traffic congestion threshold, which is used to determine if the resulting traffic at intersection of multiple roads would result in bad traffic.

On the other hand, the traffic density already calculated using wireless polling, or sensors in a future design would be communicated between nearby traffic controllers which would then be combined and compared against the threshold, and if necessary the required alerts are sent to corresponding vehicles under a determined traffic controller, and asked to divert to another road with less traffic.

After determining the threshold and the traffic density, the system diverts the traffic based on these values, and also notifies the central traffic management system of the traffic status, and current decisions it is making regarding traffic. The system is sampled every 3 seconds approximately, and the results are communicated every 2 seconds approximately, which is the time it takes for all the computations and communications to finish without any sync or interference issues along

with some added buffer time.

Hardware Details: Initially, we tried the ultrasonic sensor by connecting Vcc pin to power, Gnd pin to ground, and output pin directly to the port of the mote. But we realized the problems with the pin size of the sensors, the conversion and filtering logic required and need of sensors for every new vehicle added.

To solve this problem, we decided to go with more straight forward approach of radio polling and manipulating power of transmission. Since we tried to look for an alternative solution that is easier to carry out. We thought this approach would be good and hence implemented this approach, which is using just poling packet acknowledgment, ACK based timer synchronization, multi casting, filtering on receiver end, and UDP shell .

The result is that when the vehicle is nearby a traffic controller, it is able to send a acknowledgment packet, thus registering itself to that particular traffic controller. And all further messages are attached with identifiers telling who they belong to and what kind of message is that, for example, the messages could be of polling query or a response, it would be of a alert to particular vehicle or a simple update between two traffic controllers, these identifiers will be used to filter out unwanted responses or computations on the receiver part.Our problem of reading the values from sensor was solved.

To organize the structure of our design as well as minimizing the resource required, we limited ourselves in using exactly 2 roads with 1 traffic controllers each, and few vehicles on each road depending on the scenario.

Code/Design Reference: We did all of the design and implementation work from scratch, although we are aware of the probability that resources are out there with similar design, but since we had the implementation already planned out, we decided to conduct no research on the core implementation during the design process and to do everything from scratch. We did although look in to GUI implementation, ultrasonic sensor implementation and topics of that nature.

Chapter 4

Difficulties

Our original design was to use distance finding sensors to test the existence and then distance of other vehicles nearby. The first choice of our design was using ultrasonic distance sensors, but since we needed a minimal of 4 sensors to have a realistic detection of traffic and needed completely new logical implementation for filtering out duplicate detection and interference, we decided to go look for other ways.

We then come to think of using the IR sensor, the second best choice as it only requires an object to be close enough to activate the sensor output. However we couldn't get hold of any IR sensors, to test and try, so we were left with methods using no sensors.

With some luck we are able to figure out a way to detect and count the vehicles, and also the implementation required no external sensors, but we had to still find a way to control the power of transmission dynamically in one of our implementation, or at least statically in an another approach.

Also we had to sync the traffic report with main traffic management center, other traffic controllers, and also poll the vehicles with ACK counting and identifier based filtering mechanism, and then send alerts to certain vehicles, all in a same minimal time frame, making the probability of packet lose or a interference causing wrong values higher. We had to come up with a time synchronization mechanism to handle this without issues.

We also had issues while trying to implement a python GUI for central traffic management system, and implementing a serial forwarding mechanism, which was later modified to serial snooping mechanism on a PPP router or a base station.

Chapter 5

Result

Testing Results: We had few problems while testing too, since the motes had to be connected via USB connectors to have constant power supply, we couldn't have many vehicles and also move around vehicles and traffic controllers much, we solved this using batteries and portable power supplies, but few motes like PPP or base station router still had to be connected to a serial port for monitoring,and we couldn't do the the testing continuously with mobile power supply because of unreliability of getting a constant power supply, the mobile power supply would stop driving our mote with lower power demands as the power supplies stop detecting the mote.

We also had challenges determining the right power of transmission from vehicles, so that we would have enough for transmission to a traffic controller nearby but not enough to reach other traffic controllers, but that meant sometimes we would move a traffic controller or vehicle out of range while testing many vehicles and miss a packet or two, and have a delay in registering or alerting the vehicle in question.

After getting through these various problems, we are able to conduct our testing finally with multiple motes powered by batteries, there are total of 12 different combinations of traffic condition on which we were checking the results on: Table 5.1

Since the traffic management is based on prediction system, the traffic diverted will be from lower traffic areas moving to higher traffic areas as they have no clue of incoming traffic, and we would try to alert just enough cars above the traffic threshold level. The higher traffic areas already know they have more traffic and high probability of higher traffic in next junction, and they would follow just regular traffic management system or self decision to divert.

Table 5.1: Scenarios for traffic diversion

Road 1 Condition	Road 2 Condition	Predicted Traffic	Vehicle Diverted
HT: High Traffic	**MT: Medium Traffic**	**LT: Low Traffic Density**	**V.RX: Vehicle from Road X**
HT	HT	HT	V.R1 and V.R2
HT	MT	HT	V.R2
HT	LT	HT	V.R2
MT	MT	MT	None
MT	MT	HT	V.R1 (R1 Traffic < R2 Trf.)
MT	LT	MT	None
MT	LT	HT	V.R2
MT	HT	HT	V.R1
LT	LT	LT	None
LT	MT	MT	None
LT	MT	HT	V.R1
LT	MT	HT	V.R1

We tested out each configuration by creating different traffic conditions on our own using the Telosb motes, the result were quite reasonable and we are quite satisfied with the results of our testing. We repeated the testing several times to validate the correctness of our implementation.

We also tested for different the traffic threshold values and vehicle density quite extensively, and we could get the results and even send results to other traffic controllers consistently and also alerting the vehicles there too. The results we confirmed using monitoring python tools to see if settings and results are updated everywhere and also if results are in relation to the current traffic condition.

Speed of Execution: There's no problem within this category as the the implementation is given enough time to complete before next sampling and each sampling in itself happens very fast.

Accuracy: We couldn't do exhaustive testing on the design due to the limited timing. But in scenarios we tested we were able to get reasonable outputs.

Safety: There should be no safety issues involved in this design since there's no such parts in the system which could induce injury while operating.

Interference with Other People's Designs: We did use a separate channel, and did not encounter any problem of this type in our design, and we believe that we

are safe from this type of problem.

Usability: We believe it is actually possible to use our implementation in a real life scenario with some additions and modification given the relative separateness of the traffic management from all other environment variables. With an addition of the module to also consider the conditions at road we are diverting to, and details on the responses of the vehicles to the alert which was already sent, the system should be able to correct itself and it shouldn't be causing more traffic congestions due to, for example, any bugs in the coding or design of the system. In reality, in order to prove its true correctness and efficiency, we must apply the system in a real life situation and see how the implementation works with such environment.

Chapter 6

Conclusion

Analysis: After testing and simulation, the results met our expectations very well. When we simulated the low traffic, we moved the cars occasionally on the road. When we simulate medium traffic, we move the cars more frequently and made a car queue on the road. When we simulate heavy traffic, we move the cars at high frequency and made a constant car queue. These traffic situations simulate the real world traffic fairly well.

We believe now that, Smart Prediction Based Traffic Detection can lead to reduction in traffic congestion and achieve traffic fluency. And also automated systems lower the risk of human errors, and as a result, number of accidents may be reduced considerably.

Also, if we were to do the project next time, we would include more cases that distinguish different traffic situations. Currently, our traffic detection algorithm relies on detecting the number of vehicles passed by, and if there are any long vehicles queues. In order to distinguish more traffic conditions, we need sensors to reduce the communications from polling and the resulting interference and sync problems and probably different kinds of sensors, along with more sophisticated algorithm to handle even more scenarios and alert only particular vehicles.

In our design plan, we planned to use Ultrasonic, or IR sensors as well. However, we couldn't and we had to finish the project with an alternative, radio polling mechanism. Next time, we could use the ultrasonic, or IR sensors instead so that the model is more realistic, and we can effectively detect how long the car queues are.

We would also adjust the timing of the traffic density changes differently. Currently, we detect the traffic condition and send the alerts with fixed time intervals.

Next time, we could adjust the time intervals relatively according to traffic conditions.

Also, we could also expand the model by adding more roads in an intersection and parallel roads and more traffic controllers . In addition, we would implement more functionality such as the alert to turn left, right or back and detect cars that ignore the alerts and adjust our next alert accordingly.

Intellectual Property Considerations:

- No code/design reuse in our case, the design process is completed without any research, although there's possibility that similar design are out there on the Internet. The code is build from scratch by us, using the knowledge we obtained from Ad-hoc and sensor networks course and lab.
- No code from the public domain is reused in our design, as state above, the software is entirely coded from scratch.
- No reverse engineering in our design, we used simple modules and put everything together ourselves.
- No sample parts used in our design, we got the parts from the professor.
- No patent opportunity, as we both understand that similar designs already exists, ours is only one of the many different designs exists and thus not likely to be patentable.

Ethical Considerations: We found that our design project is related to the following points in the Code of Ethics:

1. To accept responsibility in making engineering decisions consistent with the safety, health and welfare of the public, and to disclose promptly factors that might endanger the public or the environment.

2. To be honest and realistic in stating claims or estimates based on available data.

3. To improve the understanding of technology, its appropriate application, and potential consequences.

4. To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others.

5. To assist colleagues and co-workers in their professional development and to support them in following this code of ethics.

Our project simulates the prediction based traffic management system in an intersection. This system, if made, may be used in public and has a good probability of affecting the day to day lives of large group of people. Therefore, we have to be careful in the design and accept responsibility in making engineering decisions. This project is to improve the traffic condition by managing the traffic and hence benefits the public.

We are honest with the testing results. The results are realistic that the side with more traffic would have noticed high traffic and has less space to divert until they reach intersection, but the road with less traffic currently can change their routes before coming in contact with high traffic, and has less chance of confusion and chaos due to alert in lesser traffic area. The timing of the traffic alerts is heuristic and there is some estimation involved.

Our project intends to explore the improvement of the traffic conditions with prediction system. This project uses radio polling to detect traffic and then control the traffic density accordingly. Our model can build up incrementally by using other traffic controllers, adding more intersections, and more sophisticated algorithms. When we are building the model, we thought about how to break the traffic in to modules. We looked for advices on how to model an intersection and handle power of transmission as per our needs. We got inputs from the professor and our lab assistant. The advise were on how realistic the model should be and how the working to be close to a real world intersection, the sensors that can be used, and the working of power management.

During the project, we have helped each other in our professional development by learning from each other. Often there is something that one partner knows and the other does not. We have also supported each other in following above code of ethics.

In case you have any suggestions, please do let us know. My contact details are below.

Anirudh Jayaram Shetty
shetty@campus.tu-berlin.de

Chapter 7

Tasks List And Distribution

This gives a general idea of who worked mostly on a particular task, but as with all projects, everyone has always contributed at least something tiny to a individual task.

1. Project selection and evaluation - Anirudh, Sarjo, Yatindra.
2. Sensors search and research - Sarjo
3. Short presentation - Sarjo, Yatindra
4. Defining algorithm and architecture - Anirudh, Yatindra
5. Code structure design - Anirudh
6. Coding and documentation - Anirudh, Yatindra
7. Tx power and serial snooping - Anirudh
8. Parts testing - Sarjo, Yatindra
9. Building Model - Anirudh, Yatindra
10. Implementation of python tools - Yatindra
11. Testing and debugging - Anirudh, Yatindra
12. Evaluating and building GUI model - Sarjo
13. Final testing - Yatindra
14. Final presentation - Anirudh
15. Project report and documentation - Anirudh

Chapter 8

References

1. TelosB Data Sheet

https://isis.tu-berlin.de/pluginfile.php/558248/course/section/120904/telosb_datasheet_rev20111109.pd

2. TmoteSky Data Sheet and Schematics

https://isis.tu-berlin.de/pluginfile.php/558248/course/section/120904/tmote-sky-datasheet.pdf

3. TinyOS

http://www.capsil.org/capsilwiki/index.php/TinyOS

4 Report Structure

https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects

Chapter 9

Appendix

Code Listing: Due to the long length of the code involved, it's not included as part of the report, but as a separate folder in submission also the same code can be found at git repository link:

https://github.com/AnirudhShetty/SNL_Group_3

At the time of writing this report, there were no videos or pictures taken yet, and if any of those are acquired it will be at same git-hub link given above.

Block Diagrams: Schematics of some of the hardwares used.

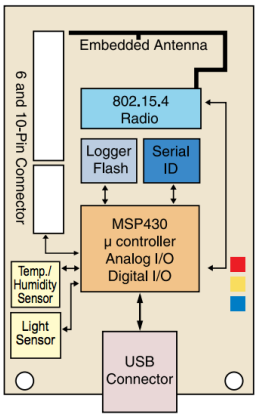

Figure 9.1: TPR2420CA Block Diagram

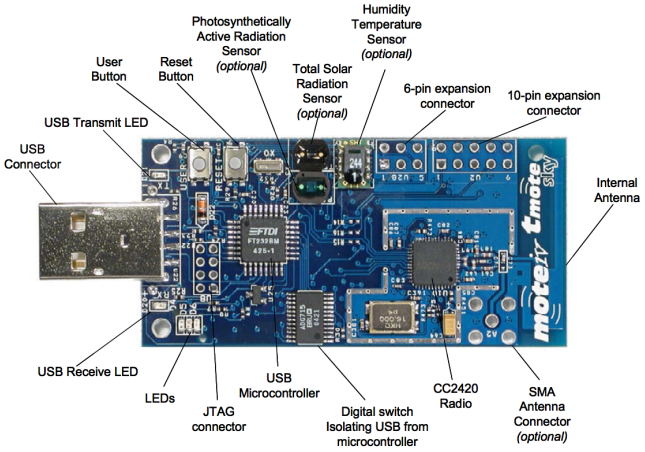

Figure 9.2: Tmote Sky module