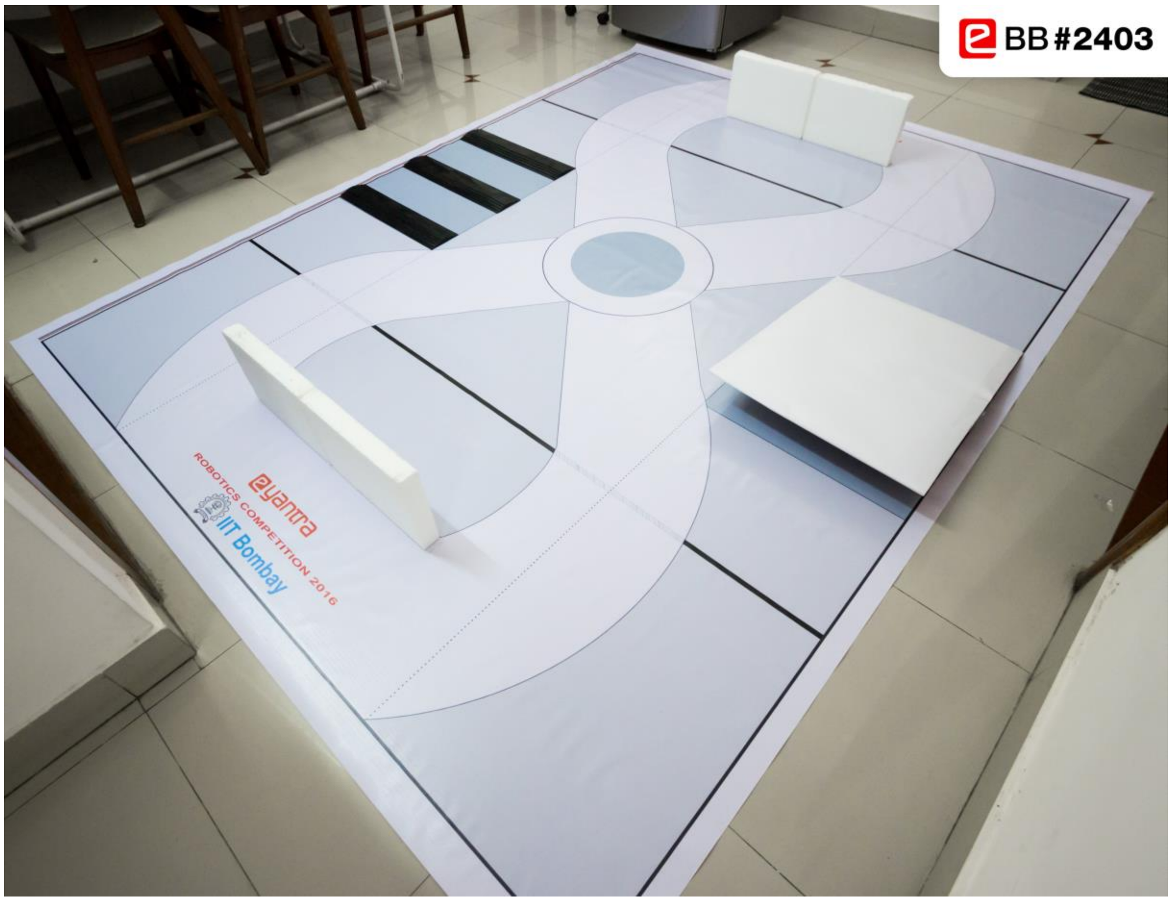
e-Yantra Robotics Competition - 2016

Theme Analysis and Implementation - Balance Bot
eYRC-BB#2403

Team Leader Name	**Heethesh Vhavle**
College	**B.M.S. College of Engineering**
E-mail	**heetheshvn@yahoo.com**
Date	**02-02-2017**

Preparing the Arena

Q1. Prepare the Arena for Balance bot and insert the image of arena here. (5)



Rules and Scoring

Q2. What are the various locations in arena where bot has to be placed for repositioning? (2)

As per the Rulebook, when the robot is performing a maneuver and it falls, reposition of the robot is allowed at certain positions for different maneuvers. For Maneuver 1 and 2, the reposition location is point 5 as labeled (Landing Zone). For Maneuver 3, the reposition location is either point 1 or 3 based on from which direction the maneuver had started. Similarly, for Maneuver 4, the reposition location is point 2 or 4 based on the start point of the maneuver.

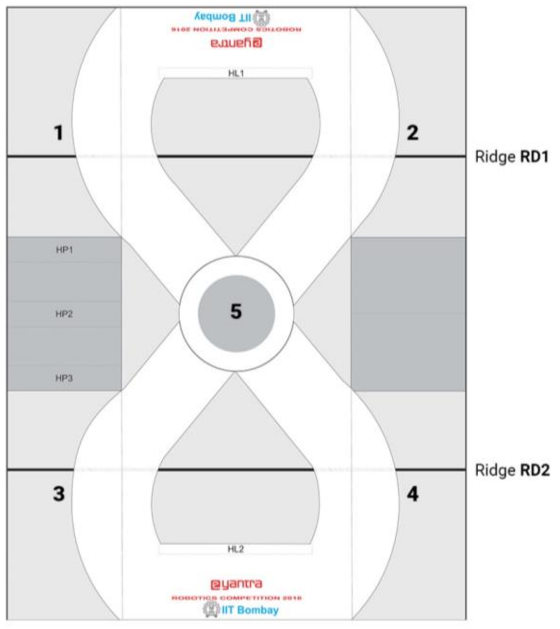


Figure 1.1: Diagram showing the location of repositions for the different maneuvers.

Q3. Balance Bot theme consists of exponential formulas for scoring. What is the combination of sequence you choose for maximum score? (3)

To optimize for best score, there are number of constraints involved such as number of maneuvers, length of the maneuvers (in terms of time and distance taken) and the risk, of the robot falling, involved with that maneuver. To maximum the score, it is necessary to complete all the maneuvers and avoid all obstacles for the bonus points. The maximum possible score is 100 (M1) + 200 (M2) + 400 (M3) + 500 (M4) + 100 (B) + 100 (Z) + 0 (HP) + 0 (FP) = ~1400 (Best Case Scenario, rounding off).

The outcome of the exponential scoring is such that, only 2-3 maneuvers of each type is required to be performed in order to get around 94% of the maximum score. As we tend to achieve a score of 1400, even more number of maneuvers is to be performed (Total of 5-6 of each type). The main concern with increased number of maneuvers and time spent on the arena is that we run a greater risk of the robot falling. If the robot falls, the bonus point will not be given and an additional fall penalty is incurred. As the number of falls increase, penalty increases exponentially and this will drastically reduce our score. The worst scenario would be the one in which the robot completes 0 maneuvers and falls down near the blocks HL1 and HL2 multiple times. In this case, M1-M4 is 0, Bonus is 0 and FP will be a large negative value and so will be the total score. For HP and FP = 10, final score will be around -7470.

M1 and M2 have the least risk involved and M4 has the greatest risk of falling involved. M1 takes the shortest time to complete, M2 takes the longest, and M3 & M4 take roughly around the same time. The order is also important and should be such that end and start of maneuvers should be very close.

Our approach to maximize the score is to first achieve around 93% score (*Safe Score*), and then aim for the 98% (*Best Score*) and then try for almost ~100% of the score (*Max Score*) and complete more number of maneuvers. Our strategy is to achieve *Safe Score* in first run and *Best Score* in second run. Penalties must be avoided at all times and is considered 0. The following table summarizes various approaches.

Strategy	**n(M1)**	**n(M2)**	**n(M3)**	**n(M4)**	**Bonus**	**Score**	**Percentage**
Safe	4	3	2	2	200	1304*	93.14%
Best	4	4	4	4	200	1380*	98.57%
Max	5	6	6	7	200	~1400*	~100%

**Scores are rounded off to nearest greatest number.*

Balance Bot Design

Q4. What is the position of CG of your Balance Bot design? Explain the pros and cons for such design. (4)

Initially, during design analysis, the ideal position of the CG was planned to be kept high. However, while balancing it, this posed a problem of limiting the maximum recoverable tilt angle of our robot. When the robot used to tilt beyond a certain angle, even at full speed of motors, the robot was not able to recover and used to topple over. The CG was then lowered by replacing the metal rods with 3D printed plastic frame and by shifting the position of the battery from the top tier to the bottom-most tier.

As of now, the CG of robot is approximately slightly above the bottom-most tier. The advantage of this, as mentioned earlier, is better stability and greater recoverable tilt angle. As we know, for a high CG, the robot tilts much slower and thus giving the motors more time to recover and hence improved response of the system, but only up to certain tilt angle. Now that we have lowered our CG, responses of the system will be slightly slow and can be considered as a disadvantage. Therefore, choosing the position of CG is a is a trade-off between slower angular acceleration and maximum recoverable angle of the bot and must be carefully chosen, by trial and error, so that robot is stable while static, in motion, on slope and uneven surfaces as well.

Q5. Constructing a balance bot in itself is a huge challenge. One of it is distributing the weights equally for stable bot. How did you distribute weights in your bot? What are the various challenges you guys faced while designing the physical structure of the robot. (3 + 3)

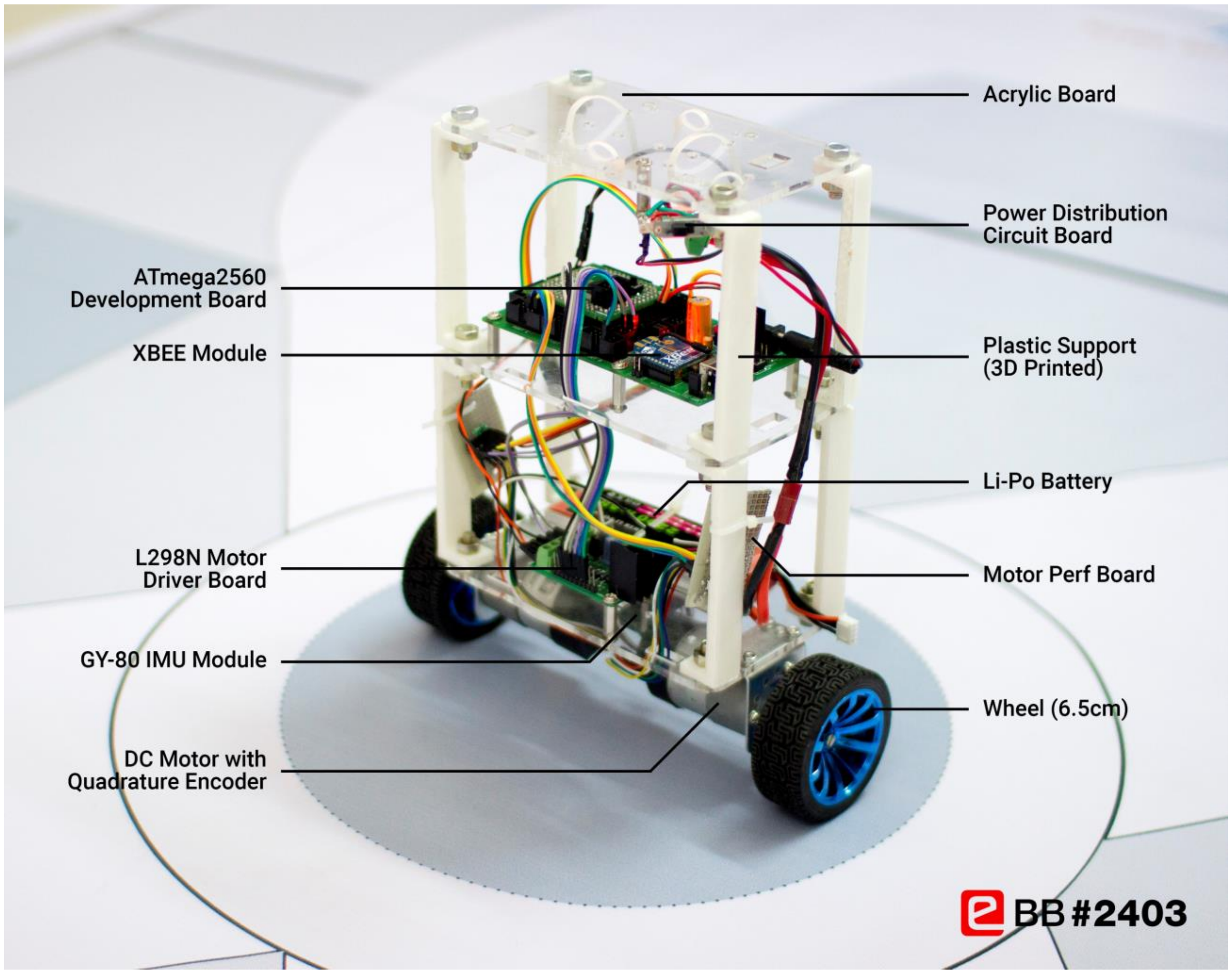

Acrylic Board
Power Distribution Circuit Board
ATmega2560 Development Board
XBEE Module
Plastic Support (3D Printed)
Li-Po Battery
L298N Motor Driver Board
Motor Perf Board
GY-80 IMU Module
Wheel (6.5cm)
DC Motor with Quadrature Encoder
BB #2403

Figure 5.1: Labeled diagram of the Balance Bot showing the placement of the various components.

In order to distribute the weights equally, all the components had to placed precisely along the center of X and Y axes. The acrylic boards were precisely designed and were laser cut to achieve this. However, the weights were not perfectly balanced and there was a slightly offset in the tilt angle (the angle at which the robot perfectly balances, when switched off). This offset angle was compensated in the software. The precise designing for laser cutting was in itself a challenge.

Another challenge that we faced was heavy weight of the metal rods we used which made the robot topple more quickly. The support frame was made lighter by using 3D printed components.

The other challenge we faced was finding the best height for the robot. A small height made the robot very unstable whereas a greater height reduced the maximum tilt angle and increased the weight of the robot. So choosing the best position of the CG, height of the robot and constantly making changes in design was a challenge.

Balance Bot Control

Q6. For controlling balance bot wirelessly, you have to design a joystick. How are you going to design a joystick? Explain the hardware and software used for construction of joystick. (10)

Hardware Design

The main components of our joystick controller include the XBEE S2C Module, 2-axis Joystick Module (2 analog inputs), 4 Push Buttons (4 digital inputs) and Power/Status Indicator LEDs (5). The power supply is provided by a single 3.7V Li-Po cell. The on-board 3.3V regulator on the XBEE adaptor can be used for this purpose. However, the AMS1117 3.3V regulator requires an input voltage greater than 4.8V for this. Thus, a DC-DC boost module is used to boost the 3.7V to a voltage greater than 5V. Auxiliary USB power supply can also be used for practicing in place of battery.

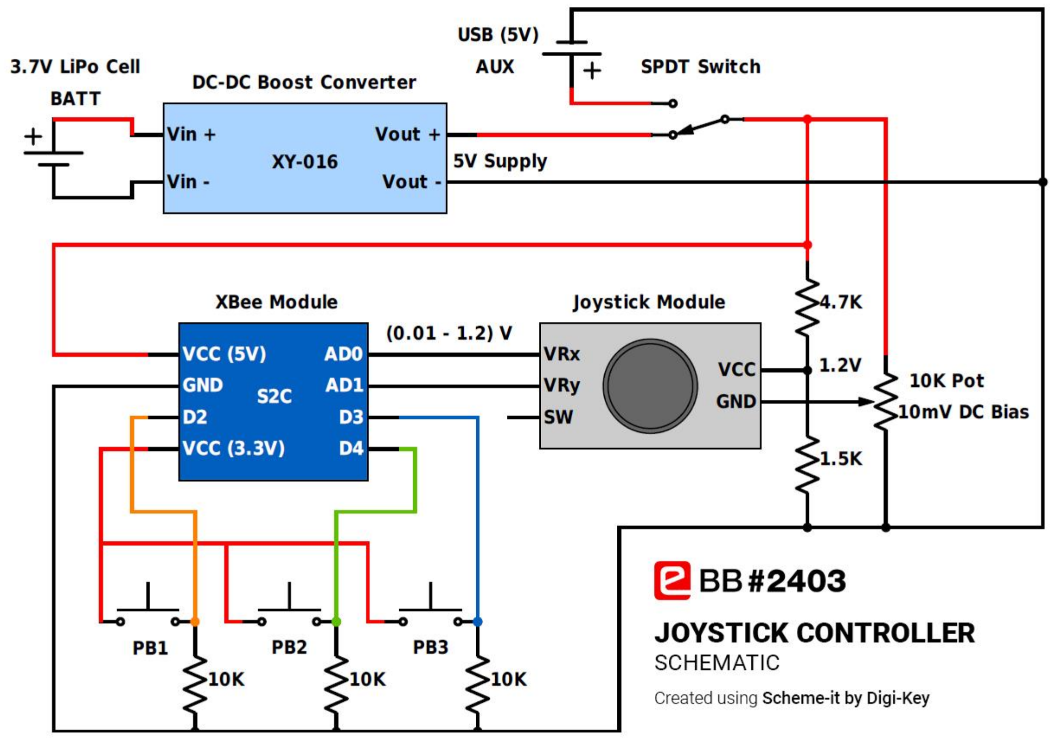
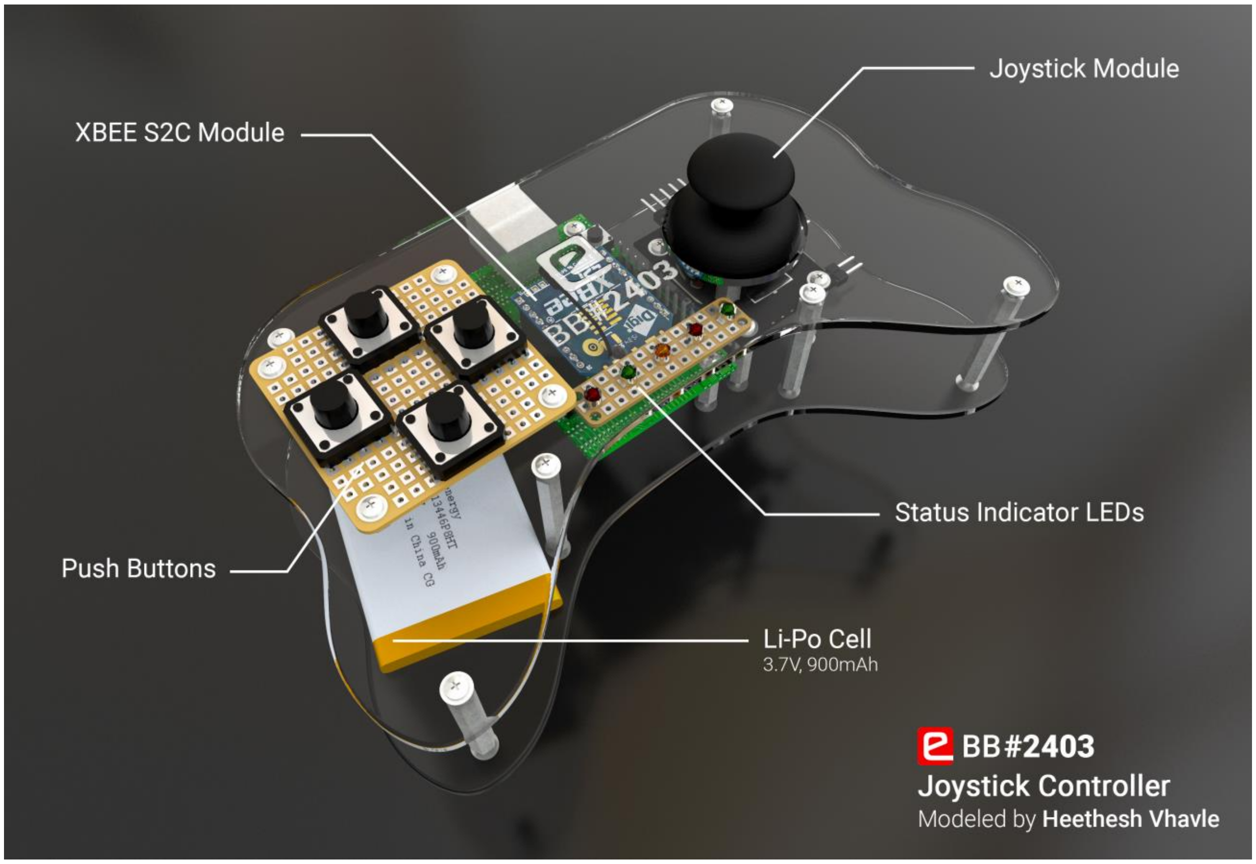


Figure 6.1: Schematic of the Joystick Controller (**Push Button 4 is not shown, connected similarly to D11 on XBEE*).

The main goal of the hardware design is to keep it as small as possible and minimize the power consumption and is one of the reasons we avoided bigger batteries and low efficiency linear regulators. Additional push buttons are included to provide more functionality in the future (such as multiplexing them for on-the-fly PID tuning or different types of turns and rotations). The main use of it is to sound the buzzer as and when required. The reference voltage of ADC in XBEE is fixed at 1.2V. Therefore, a potential divider is first used to provide 1.2V for the joystick module VCC. The ADC values were incorrect for the range 0-5mV. To solve this problem, the input range to the ADC pins was shifted to 10mv – 1.2V by providing a 10mV DC bias to the GND of the joystick module, using a simple voltage divider. Thus, a

0V is never required to be read by the ADC pins, which solves our problem. 3D representation of the controller is shown below.



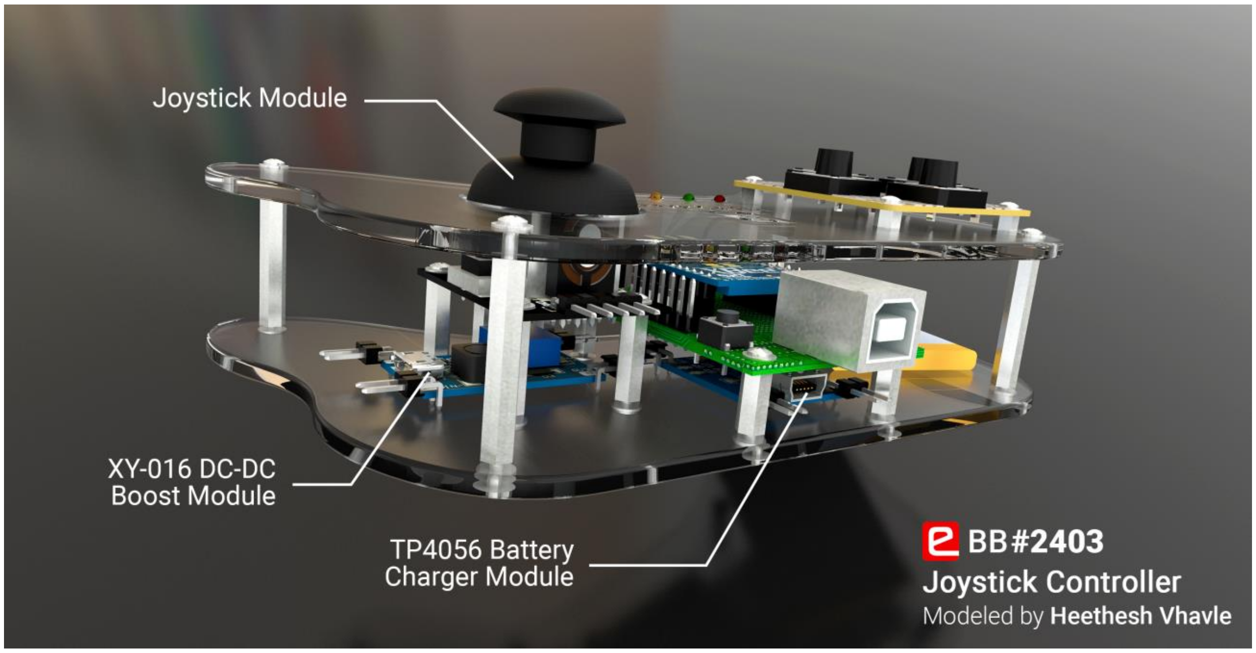

Joystick Module
XY-016 DC-DC Boost Module
TP4056 Battery Charger Module
BB#2403
Joystick Controller
Modeled by Heethesh Vhavle

Figure 6.2: 3D Model of the Joystick Controller showing the position of various components.

Software Design

The XBEE module mounted on the robot is configured as the Coordinator in API mode. The XBEE module on the controller is configured as End Device in AT mode and is responsible for sending the data frame. Let the data frame consists of the 1 digital and 2 analog (ADC) data samples.

```
7E 00 0E 83 00 00 22 00 01 06 10 00 00 02 0C 02 0C 27
```

Start delimiter: 7E
Length: 00 0E (14)
Frame type: 83 (RX (Receive) Packet 16-bit Address IO)
16-bit source address: 00 00
RSSI: 22
Options: 00
Number of samples: 01

Digital channel mask: 00 10
Analog channel mask: 06 00
DIO4/AD4 digital value: Low
DIO0/AD0 analog value: 02 0C (524)
DIO1/AD1 analog value: 02 0C (524)
Checksum: 27

The sampling rate is set at 50ms. For the coordinator XBEE, a task can be scheduled on the microcontroller, to read this data frame every 50ms (or UART signal interrupt can be used). First, the start delimiter has to be compared with the value 0x7E. Once a match is found, the next 11 bytes can be read and discarded. The 13th byte will give the status of the digital pin and the pair of next 2 bytes each will give the 10-bit ADC data ranging from 0x00 (0V) to 0x3FF (1.2V) and then clear the serial buffer.

Once we have the data from the joystick module, the position set point variable (for the PID loop) can be updated to achieve forward and backward motion. For left and right rotation, the PWM offset variables can be updated. Similarly, a buzzer state variable can be set to HIGH to sound the buzzer, once a 0x10 is read. The buzzer can be automatically switched off after a set time period. Similarly the other push buttons can be configured for various other functionalities.

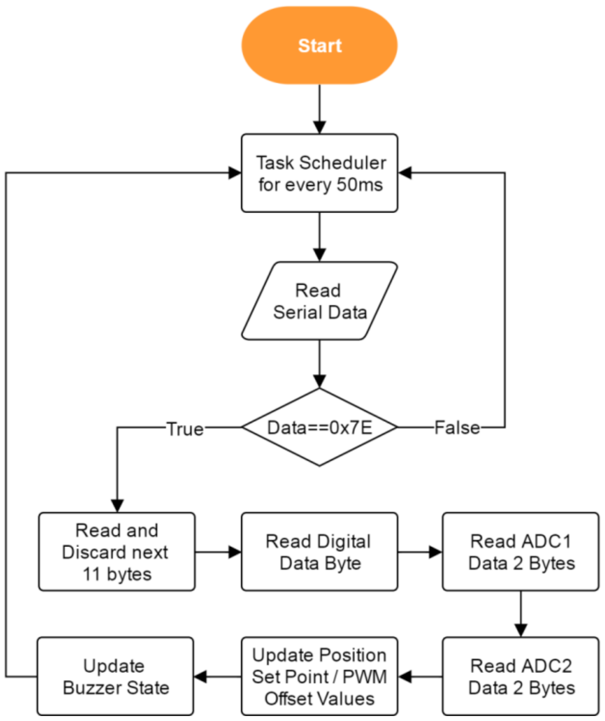


Figure 6.3: Flowchart representing the microcontroller code for the wireless Joystick Controller.

Theme Implementation

With the end of Task 3 you must have finished the construction of the balance bot. Now, study the rulebook and try to implement the solution for the theme. Final task of the competition is to submit the video and code for the theme implementation. Submission date for video and code will be notified in future.

Start early and be ready for video submission. Stay tuned for further instruction.