Software Team Positions

Last Updated: 01/16/2016

|  |  |
| --- | --- |
| Roles | Contributors |
| Lead Programmer | Bryan Bettis  Brian Sadowitz (“In-Training”) |
| Devices | Adam Sweiger  Bryan Bettis |
| TeleOp | Jack Studier  Luke Herbert |
| Auto – Task Control | Bryan Bettis  Holly Langenstein (Now primarily hardware role) |
| Auto - Navigation | Eamonn Ryan  (TBD?) |
| Auto - Actions | Brian Sadowitz  Jack Studier |

Software Role Descriptions

|  |  |
| --- | --- |
| Role | Opportunities/Responsibilities |
| Lead Programmer | A lead programmer is one of the more experienced software developers on the team. They provide software-related assistance and training to other members as needed. They also help to manage software-related information, tasks, deadlines, and so on that is/are used by the other programmers. |
| Devices | The devices programmers communicate with the hardware designers to write the classes that represent physical mechanisms and electronics on the robot. This code will be used by all other classes as a standard way to access and test these physical components. Device classes can represent something as simple as a motor or something more complex like the entire drive system (which is a group of motors and other classes). |
| TeleOp | The TeleOp programmers write the classes dealing with the tele-operated phase of an FTC match. They work with the primary drivers to setup the controls for the device interfaces created by the device programmers. They not only write the controls for the robot mechanisms, they can also write code automating certain actions, like emergency stops preventing the robot from hurting itself, or like setting maximum positions on mechanisms. |
| Auto – Task Control | The task control programmers write the classes managing the flow of robot actions and events during the autonomous phase. Some of these classes provide a framework used by other autonomous classes to integrate with the rest of the autonomous program. These classes do not directly affect how these actions or events are completed, they control the order they happen in. The task control programmers also manage all the different autonomous opmodes. |
| Auto – Navigation | The autonomous navigation programmers write the classes that handle navigating the robot around on the field. They use sensor information to get the robot to move from point A to point B, or to face a certain direction. These programmers will use various techniques to move the robot around the field easily and consistently, like grid systems, simple or complex course correction, or calculating distances and headings. The task control classes say where to move the robot, and the navigation classes figure out how to get there. |
| Auto – Actions | The autonomous actions programmers work with the sensors and manipulators on the robot to accomplish different tasks. An “action event” can be as simple as moving a servo to a position, or it can be a combination of different actions, like taking color sensor readings and then pressing the right button on the beacon. An action could even tell the navigation classes to move to a different position based on sensor readings, like moving to the right or left beacon button. Task control tells the action classes what “action event” to do, the action classes figure out how to do the “action event.” |