# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>1</td>
</tr>
<tr>
<td>Function</td>
<td>1</td>
</tr>
<tr>
<td>Platform</td>
<td>1</td>
</tr>
<tr>
<td>Customers</td>
<td>1</td>
</tr>
<tr>
<td>Goals and Scope</td>
<td>2</td>
</tr>
<tr>
<td>Included</td>
<td>2</td>
</tr>
<tr>
<td>Excluded</td>
<td>2</td>
</tr>
<tr>
<td>Deliverables from Project Concepts and Efforts</td>
<td>3</td>
</tr>
<tr>
<td>Common</td>
<td>3</td>
</tr>
<tr>
<td>Model</td>
<td>3</td>
</tr>
<tr>
<td>Platform</td>
<td>3</td>
</tr>
<tr>
<td>Major Department Deliverables</td>
<td>4</td>
</tr>
<tr>
<td>Risk Management</td>
<td>5</td>
</tr>
<tr>
<td>Work Breakdown</td>
<td>7</td>
</tr>
<tr>
<td>Cycle 4 (weeks 1-2)</td>
<td>7</td>
</tr>
<tr>
<td>Cycle 5 (weeks 3-4)</td>
<td>8</td>
</tr>
<tr>
<td>Cycle 6 (weeks 5-6)</td>
<td>10</td>
</tr>
<tr>
<td>Cycle 7 (weeks 7-8)</td>
<td>10</td>
</tr>
<tr>
<td>Cycle 8 (weeks 9-10)</td>
<td>10</td>
</tr>
<tr>
<td>Finals week</td>
<td>10</td>
</tr>
<tr>
<td>Metrics Strategy</td>
<td>11</td>
</tr>
<tr>
<td>Purpose</td>
<td>11</td>
</tr>
<tr>
<td>GQM Outline</td>
<td>11</td>
</tr>
<tr>
<td>Metric 1</td>
<td>11</td>
</tr>
<tr>
<td>Metric 2</td>
<td>11</td>
</tr>
<tr>
<td>Metric 3</td>
<td>11</td>
</tr>
<tr>
<td>Technical Process</td>
<td>12</td>
</tr>
</tbody>
</table>
Overview

Function

The platform sub-project is divided as follows:

- Microcontroller software to control the robot hardware

- Language bindings to control the microcontroller software through a standard application programming interface. Motion commands and diagnostic queries are sent from a program on the PC to the microcontroller software on the robot.

- Robot control applications with textual and graphical interfaces

The model sub-project consists of a model of the RP10 Robotics Platform in Microsoft Robotics Developer Studio to accurately simulate the real-world interaction of the robot in a virtual three-dimensional setting.

Platform

The following platforms are targeted:

- Freescale MC9S12DT256 microcontroller for on-board robot software

- Windows NT family operating system with .NET Framework for initial API binding and control applications and with Microsoft Robotics Developer Studio for model simulation

Customers

Our customer is the Department of Mechanical Engineering at the RIT Kate Gleason College of Engineering. Our contact providing us with direction and details for the project is Prof. Wayne Walter.
# Goals and Scope

The initial scope focuses only on core functionality of the robot platform across the RP10 and the PC. See risks for justification.

## Included

<table>
<thead>
<tr>
<th>Goal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API Bindings</td>
<td>PC software will interact with the robot through an API. Microsoft's .NET Framework is the first technology chosen to implement the API.</td>
</tr>
<tr>
<td>Control Application</td>
<td>Applications will be built on top of the API for manual robot control. The user will interact with the application through text commands (i.e., console) or a graphical user interface.</td>
</tr>
<tr>
<td>Serial Cable Communication</td>
<td>The robot will communicate with the host PC by wire connected to a serial port.</td>
</tr>
<tr>
<td>User manuals</td>
<td>User manuals will be written to describe usage of control applications and development with the API bindings.</td>
</tr>
<tr>
<td>Motor variability</td>
<td>A goal of the RP10 is to support 1 to 4 motor modules.</td>
</tr>
<tr>
<td>Model</td>
<td>The software model of the RP10 in MRDS for simulation.</td>
</tr>
</tbody>
</table>

## Excluded

The following table describes functionality that are either simply out of scope permanently or are potentially in scope in the future.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB Binding</td>
<td>Although MATLAB is a prominent engineering tool, tool knowledge will be required. The team's mechanical engineers will be consulted.</td>
</tr>
<tr>
<td>Wireless Robot Control</td>
<td>The hardware will not be available to do this.</td>
</tr>
<tr>
<td>Payloads</td>
<td>Although the RP10 is designed to carry up to 10 kg, payloads will not be factored due to the absence of motion sensors.</td>
</tr>
<tr>
<td>Robot Applications</td>
<td>The platform enables engineers and scientists to develop applications for the robot such as sensor networks.</td>
</tr>
<tr>
<td>Navigation</td>
<td>Motion sensors are necessary to verify navigation (i.e., move a specific distance). They are not available.</td>
</tr>
<tr>
<td>Joystick</td>
<td>We do not have a joystick. Even if we did, this would add significant research into hardware interface libraries.</td>
</tr>
</tbody>
</table>
Deliverables from Project Concepts and Efforts

In the project concepts for model and platform projects, several deliverables were listed. Several implied products have also been added below. Note that these are expected deliverables and might not match the final deliverables, depending on project progress.

**Common**

1. All documents generated by the project shall be posted on the EDGE website.

**Model**

2. Documentation of the platform characteristics and model

3. Design documentation for the platform model implementation and real platform interface

4. User manual describing how to use the platform model and real platform interface within the Microsoft Robotics Studio (MRS).

5. A demonstration of the software model and robotics applications developed under the MRS. A demonstration of the platform executing a program running in MRS.

6. A Solidworks 3-D model of the robot with physical and material properties

7. Model simulation implementation that integrates with the Microsoft Robotics Developer Studio frameworks and platform API

8. Simulink model of the platform, a basis for the kinematic motion described in the platform characteristics document

**Platform**

2. Documentation of the platform software requirements

3. Design documentation for the platform software and the RP10 application programming interface

4. User manual describing how to use the platform software and the RP10 application programming interface

5. A demonstration of a text-based and GUI application to control and monitor the RP10 platform operations

6. Microcontroller software implementation

7. API implementation for at least one binding that supports communication with the microcontroller
## Major Department Deliverables

This list omits items such as peer evaluations that are required but too small to be of concern. See the official project timeline in the SE Senior Project Community at [Mycourses](https://mycourses) for these small items. Deliverables are due by the end of the associated week.

<table>
<thead>
<tr>
<th>Week</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Updated project plan</td>
</tr>
<tr>
<td>5</td>
<td>Project poster concept</td>
</tr>
<tr>
<td>6</td>
<td>Preliminary project poster</td>
</tr>
<tr>
<td>7</td>
<td>Project poster</td>
</tr>
<tr>
<td>8</td>
<td>Draft final presentation</td>
</tr>
<tr>
<td>8</td>
<td>Technical report outline</td>
</tr>
<tr>
<td>9/10</td>
<td>Final presentation</td>
</tr>
<tr>
<td>10</td>
<td>Draft technical report</td>
</tr>
<tr>
<td>10</td>
<td>Team final self-assessment</td>
</tr>
<tr>
<td>10</td>
<td>Post-Mortem Curriculum Reflection</td>
</tr>
<tr>
<td>10</td>
<td>Project website and repository (*.se.rit.edu) up-to-date</td>
</tr>
<tr>
<td>11</td>
<td>Final technical report</td>
</tr>
<tr>
<td>11</td>
<td>Summary of Curriculum Reflection</td>
</tr>
<tr>
<td>11</td>
<td>Senior survey</td>
</tr>
</tbody>
</table>

Weekly delivery: team member time tracking, metrics tracking (see metrics strategy)
## Risk Management

The following risks are listed in order of descending priority, determined by a combination of probability and impact (low, medium, high).

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Probability</th>
<th>Impact</th>
<th>Mitigation/Recovery Strategy</th>
<th>Trainwreck Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Drive cannot be demonstrated due to hardware problems.</td>
<td>high</td>
<td>high</td>
<td>Document known issues extensively with workarounds; talk to previous team EE lead, Walter, Vallino</td>
<td>End week 7</td>
</tr>
<tr>
<td>2</td>
<td>Services cannot represent robot parts.</td>
<td>high</td>
<td>high</td>
<td>Create a service stub to demonstrate integration with MRDS environment with visualization and physics.</td>
<td>End week 5</td>
</tr>
<tr>
<td>3</td>
<td>3-D model cannot retain mass and joint properties when imported into MRDS.</td>
<td>medium</td>
<td>high</td>
<td>Break down model into sub-models (multiple *.obj files) with their motion described in the model implementation</td>
<td>End week 6</td>
</tr>
<tr>
<td>4</td>
<td>The platform and model implementation software cannot be deployed easily to new users in a clean production environment.</td>
<td>medium</td>
<td>high</td>
<td>Create and test a procedure for packaging, deploying, and installing software with all external dependencies</td>
<td>End week 10</td>
</tr>
<tr>
<td>5</td>
<td>Progress is not clear, and schedule slips.</td>
<td>medium</td>
<td>medium</td>
<td>Track slippage and adjust task planning, review deliverables weekly.</td>
<td>N/A</td>
</tr>
<tr>
<td>6</td>
<td>Accuracy of the Simulink model for robot platform characteristics is not verifiable.</td>
<td>medium</td>
<td>low</td>
<td>Create test procedures to evaluate the real robot's kinematic motion, compare to Simulink model.</td>
<td>N/A</td>
</tr>
<tr>
<td>7</td>
<td>Services are not controllable through robot applications.</td>
<td>low</td>
<td>high</td>
<td>Create sample VPL program that controls a service stub.</td>
<td>End week 6</td>
</tr>
<tr>
<td>8</td>
<td>Model implementation is not testable.</td>
<td>low</td>
<td>high</td>
<td>Create test cases and a test application (VPL) with top-down construction of model.</td>
<td>End week 6</td>
</tr>
<tr>
<td>9</td>
<td>Platform software is not testable.</td>
<td>low</td>
<td>high</td>
<td>Decouple functionality in API at multiple levels of abstraction (client state, protocol communication, hardware control).</td>
<td>End week 7</td>
</tr>
<tr>
<td>10</td>
<td>API and MRDS model implementation cannot be integrated by design.</td>
<td>low</td>
<td>medium</td>
<td>Model developers create an integration strategy that allows MRDS services to easily switch to the real platform. API developers allow asynchronous robot control to support MRDS service-oriented architecture.</td>
<td>End week 7</td>
</tr>
<tr>
<td>11</td>
<td>Serial cable connection is unreliable for real-time robot control.</td>
<td>low</td>
<td>medium</td>
<td>Include reliability measurements in protocol for packet loss, maximize efficiency in communication.</td>
<td>End week 6</td>
</tr>
<tr>
<td></td>
<td>The robot is insufficiently powered due to uncharged batteries.</td>
<td>low</td>
<td>medium</td>
<td>Run with 2 batteries only for testing purposes and regularly charge. Charge whenever meeting in lab.</td>
<td>N/A</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>13</td>
<td>GUI control application is not intuitive or productive to use.</td>
<td>low</td>
<td>low</td>
<td>Weekly prototyping with evaluation by Walter</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Work Breakdown

Deprecated as the Trac ticket system includes the latest tasks.

Cycle 4 (weeks 1-2)

1. Project
   1. Cycle metrics data sheet (Sahil)
   2. Team document reviews (Sahil)
      1. Platform characteristics test plan
3. Platform
   1. GUI Control Application (Joe)
      1. Functional prototype
      2. RP10 motor controls through API (see all under API>>RP10-specific>>Motor Controls)
   3. Requirements
      1. User stories for control application
2. API (Karl)
   1. Functional prototype
   2. RP10-specific
      1. Motor controls
         1. Bay configuration rules
         2. Power setting (speed)
            1. Drive motor
            2. Steer motor
         3. Direction setting
            1. Drive motor
            2. Steer motor
         4. Motor stop
            1. Drive motor
            2. Steer motor
      2. Heartbeat ping
      3. Battery voltage reading
      4. Encoder reading
      5. MCU external input reading
      6. Set MCU external output
3. Protocol (Adam)
   1. Design document
   2. Support for API features (see all under API>>RP10-specific)
   3. Component status reading
      1. Motors
4. MCU (Adam)
   1. Support for API features (see all under API>>RP10-specific)
   2. Component status
      1. Motors
      3. Heartbeat acknowledge
5. Test plan (Paul)
   1. Test cases documented
      1. Coverage of each API feature above
1. Coverage of the Acceptance Test Plan
2. Coverage of the Acceptance Test Cases from the User Case
   2. Fuzz tests (throw garbage over the protocol)
   3. User stories mapped to acceptance tests
2. Strategy for developing a test harness
4. Model
   1. Crude, MRDS-importable object file format box model (Kyle)
      1. Import a crude box model into MRDS
      2. How to preserve material properties from SolidWorks to MRDS?
      3. Import a crude model with material properties to see what is retained or lost in the conversion to .obj.
      4. Is there a format specification for .obj? What does it contain?
   2. Platform characteristics test plan (Jon)
      1. Outline of procedure for capturing real platform’s acceleration and velocity
      2. What are the necessary tools?
      3. Step-by-step procedure
      4. Data analysis procedure
   3. Brick service entity (Jeff)
      1. How do services map to visual representations?
      2. Create entity source code, manifest files
   4. Updated integration strategy for MRDS-API (Sahil)
      1. Correction based on official implemented design of API
2. Department Deliverables (Sahil)
   1. Updated project plan sections
      1. Overview
      2. Goals and Scope
      3. Deliverables
      4. Risk Management
      5. Work Breakdown
      6. Metrics Strategy
      7. Technical Process
   2. Updated EDGE with important subset of wiki material
      1. What material needs to be copied to EDGE?
      2. Does one or both projects need to be updated?

**Cycle 5 (weeks 3-4)**

1. Project
   1. Team document reviews (Sahil)
      1. Platform software test plan
   2. Platform
      1. Functional specification (Sahil)
      2. Hardware problem document (Joe)
      3. Hardware problem diagnosis & resolution (Joe)
      4. GUI Control application (low priority) (Joe)
      1. Non motor-control features for RP10 (see all in Cycle 4>>Platform>>API>>RP10-specific, excluding motor controls)
      2. Updated user stories and acceptance tests
   5. API (Karl)
      1. Emergency stop
2. Heartbeat time-out handler
3. Asynchronous motor control operations for MRDS
4. Initial user manual

6. Determine second binding to implement
   1. What is the purpose of the second binding?
   2. Comparison of suitable options

7. Design document for MCU/Protocol (Adam)
8. Protocol (Adam)
   1. Packet loss handling
   2. Code documentation update
   3. Asynchronous motor control operations for MRDS

9. MCU (Adam)
   1. Heartbeat time-out
   2. Code documentation update

10. Test plan (Paul)
    1. Define test cases for the API
    2. Implement test harness as defined in previous strategy
    3. Develop test cases based on last cycle's test plan
    4. Execute all test cases and report passage

3. Model
   1. Platform characteristics (Jon)
      1. Update Simulink model with motor power variability (10-100%)
      2. Update Simulink model with motor variability (1-4)
      3. Run tests against real platform as described in test plan
      4. Record results and deposit in characteristics document
   2. 3-D Model (Kyle)
      1. Break down model into sub-models
         1. Brick
         2. Idler wheel mount
         3. Idler wheel
         4. Drive wheel mount
         5. Drive wheel
      2. Advise model developers to describe motion of sub-models in implementation
      3. Import sub-models to retain association of parts as one model
   3. Brick service module (Aaron)
      1. Create a stub service to represent a brick from scratch
      2. Demonstrate operation with VPL
         1. Create a single operation that does console output and/or returns a value
         2. Create a very small VPL application to execute operation
      3. Add stub operations for robot control based on platform API
   4. Create entity for each of Kyle's sub-model (Jeff)
   5. VPL test application
      1. How do we apply the platform test plan to the simulation? (Sahil)
      2. Make a VPL application that executes brick operations based on test cases (Jeff, Aaron)
Cycle 6 (weeks 5-6)

1. Project
   1. Platform
      1. Official design documentation for platform software and API
      2. Textual control application
         1. Determine commands
         2. Implement application
      3. API
         1. Remaining asynchronous controls
         2. Implement second binding (MATLAB, Python, etc.)
         3. Update user manual
   4. Protocol
      1. Remaining asynchronous controls
      2. Demonstration of text and GUI control applications
2. Model
   1. Official design documentation for implementation and real platform interface
   2. Usage manual for model implementation and real platform interface
   3. Demonstration of a robot application running against the software model
   4. Demonstration of platform execution of an application in MRDS

2. Department
   1. Project poster concept
   2. Preliminary project poster

Cycle 7 (weeks 7-8)

1. Department
   1. Project poster
   2. Draft final presentation
   3. Technical report outline
   4. EDGE update project

Cycle 8 (weeks 9-10)

1. Department
   1. Final presentation
   2. Artifacts CD
   3. Draft technical report
   4. Team final self-assessment
   5. Post-Mortem Curriculum Reflection
   6. Project website and repository (1..se.rit.edu) up-to-date

Finals week

1. Department
   1. Final technical report
   2. Summary of Curriculum Reflection
   3. Senior survey
Metrics Strategy

Purpose

Beyond half-baked time tracking, no metrics were recorded in the first quarter of this project. A set of metrics are provided below to support goals for progress visibility and quality assurance. The SE department requires at least 2 metrics to be recorded. The team should track at least the minimum but at most the metrics which actually deliver value. The Goal-Question-Metric (GQM) model was applied to justify the metrics.

The data for metrics is accumulated in a wiki page. At a minimum, it is updated at the end of each cycle.

GQM Outline

Metric 1

<table>
<thead>
<tr>
<th>Goal</th>
<th>Cycles should be planned with accurate estimates of the work that can be completed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>How much work was not completed that had been planned for at the beginning of the cycle?</td>
</tr>
<tr>
<td>Slippage:</td>
<td>Count the number of tickets in a cycle that were not completed by the end of the cycle</td>
</tr>
</tbody>
</table>

Metric 2

<table>
<thead>
<tr>
<th>Goal</th>
<th>At the end of each cycle, the team is aware of progress relative to the overall project.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>How much of the project has been completed?</td>
</tr>
<tr>
<td>Project completion percentage:</td>
<td>count of tickets that were resolved as fixed in cycle divided by count tickets existing</td>
</tr>
</tbody>
</table>

Metric 3

<table>
<thead>
<tr>
<th>Goal</th>
<th>The team is aware of the degree to which the current product satisfies defined tests.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question</td>
<td>How many executable test cases do the current builds pass?</td>
</tr>
<tr>
<td>Test Pass Percentage:</td>
<td>Tests passed divided by total tests</td>
</tr>
</tbody>
</table>
Technical Process

The Spiral process methodology has been applied to this project. The project timeline is divided into 2 week cycles. Each cycle starts with a review of the previous cycle, risks, and requirements. The risks dictate the work for the cycle. This methodology guides the team to manage numerous risks that still exist in the project through frequent review.

The Trac wiki contains almost all living documents, except for some in the Subversion repository which include diagrams that are incompatible with the wiki text formatting. The content is divided into cycles. When a document from a previous cycle is significantly changed in a later cycle, it is first copied into the new cycle and then changed. Thus, one can see exactly what was done in a cycle by looking at the cycle-specific page copy instead of having to browse the change history.