	Mars Rover Design Team	Team Continuum	Cornell Mars Rover	ITU Rover Team	UWRT Robotics Team	Ryerson Rams Robotics (R3)	SJSU Robotics	Team Anveshak
School Name	Missouri University of Science and Technology	University of Wroclaw, Poland	Cornell University, USA	Istanbul Technical University, Turkey	University of Waterloo, Canada	Ryerson University, Canada	San Jose State University, USA	Indian Institute of Technology, Madras
Final Score (Rank)	403.4 **(1)**	336.3 (2)	264.1 (11)	243.1 (13)	225.7 (15)	190.9 (21)	164.3 (26)	151.4 (29)
Computers on rover	Raspberry Pi, TIVA-C Connected, MSP-432, Launchpad-C2000	A Banana Pi, 3x Raspberry Pi, 1x Jetson (optionally), multiple STM microcontrollers	A Intel NUC N82E16856102053, and 8x PIC32 MX530F128H microcontroller	A Raspberry Pi 3 with 64gb SD card running Ubuntu 16.04, STM32F103 microcontrollers	A FitPC miniature fanless PC	A Jetson TX1 with 32 GB SD card, Ubuntu 16.04, and 2x Arduino Mega microcontrollers	Odroid XU4, and Teensy 3.2 microcontroller	A Thinkpad T460 laptop running Ubuntu 14.04, and Arduino microcontrollers
Joysticks	Xbox Controller, Logitech Extreme 3D Pro	Logitech Gamepads	Logitech Gamepad F310, Thrustmaster VG T16000M FCS Joystick	2x Logitech Extreme 3D Pro, one for driving and one for the arm	2x Logitech joysticks for the arm, and an Xbox controller for driving	Xbox 360 Controller for drive, Logitech Extreme 3D Pro for arm	Logitech Extreme 3D Pro Joystick	2x Logitech F310 Gamepads, one for telemetry control and one for auger/arm
Cameras	Lorex, Sony EFFIO CCD Superhead	Standard Raspberry Pi cameras and two with wide angle lenses	Logitech HD Laptop Webcam C615, x264 video encoding	5 IP cameras used for security and an Xbox 360 Kinect v1 for image processing and fake laser	2x Pointgrey cameras, 1x USB Camera	ZED depth camera, 2x BL170 degree fisheye cameras	CCD 700TVL Composite video cameras (RunCam Swift 2.0)	SJ-CAM, IP-Camera, and a Logitech webcam. Cameras were interfaced using the "motion" Linux package, though it lags and quality was not great
GPS	MTK 3339	Ublox GPS	USGlobalsat BU-353-S4	Radiolink M8N	Microstrain	Linx FM Series GPS Receiver	UBlox GPS 7	ROS All Sensors Android App
IMU	LSM9DS1	Tried multiple units, nothing really worked	SparkFun SEN-13762, chip: MPU-9250	GY-80	Microstrain	MPU-9250 module, couldn't get it working	BNO055	ROS All Sensors Android App on Moto Play G4 phone
Software Packages	Energia, TI motorware, OpenCV	ROS kinetic with joint_state_ controller, rviz, rqt, robot_localization, and more	ROS packages control-toolbox, dwa-local-planner, gazebo-ros-pkgs, gpsd-client, image-transport-plugins, image-rotate, pid, ros-controllers, spacenav-node, usb-cam, rplidar-ros, and gmapping	ROS Kinetic with packages depthimagetolaserscan, huksy_control, move_base, actionlib, cv_bridge, image_transport and more	ROS Indigo with packages socket_canbridge, rosbridge_server, teleop_twist_joy, and more	ROS Kinetic with packages rqt_image_view, rtabmap, move_base, mapviz, joy, rtimulib_ros, zed_ros_wrapper, rgbd_odometry, usb_cam, and nmea_navsat_driver	Custom framework RoverCore-S, RoverCore-F, RoverCore-MC, built in house	We used ROS Kinetic and Indigo with packages joy, rosserial, amcl, and robot_localization
Autonomous System	OpenCV, Python	Implemented on our own using GPS and distance to the goal. A control PID with some constraints and logic to back up if necessary to leads us to a given point. Goals are set when previous one was reached.	ROS move_base	ROS move_base and as a backup waypoint navigation using yaw and gps. Also, a C++ OpenCV tennis ball finding algorithm on top of ROS. We could find and navigate to the tennis ball from 8m. .	move_base and robot_ localization	ZED depth camera, rtabmap, move_base. We first teleoperate to build a SLAM map and find the tennis ball by human eye, then we go back to the start and set an autonomous goal in the SLAM map.	GPS and drive system, no need for anything else	We had plans of using AMCL and sensor fusion by making use of the existing packages in ROS, but ran out of time.
Arm Control Software	Custom in solution in Energia. interfaced with custom control software RED (Rover Engagement Display) at base station	Tried MoveIt but implemented our own	Some experiments with MoveIt inverse kinematics but used forward kinematics at competition	Wrote our own inverse kinematics and simulation in Unity using C#	Wrote our own PWM library for arm motors	We had plans to use MoveIt but due to lack of testing time used velocity control for each joint mapped to a joystick	We wrote firmware into our framework for our Teensy 3.2 MCUs	Open-loop control with commands sent to an Arduino

	Mars Rover Design Team	Team Continuum	Cornell Mars Rover	ITU Rover Team	UWRT Robotics Team	Ryerson Rams Robotics (R3)	SJSU Robotics	Team Anveshak
School Name	Missouri University of Science and Technology	University of Wroclaw, Poland	Cornell University, USA	Istanbul Technical University, Turkey	University of Waterloo, Canada	Ryerson University, Canada	San Jose State University, USA	Indian Institute of Technology, Madras
Final Score (Rank)	403.4 **(1)**	336.3 (2)	264.1 (11)	243.1 (13)	225.7 (15)	190.9 (21)	164.3 (26)	151.4 (29)
Wireless radios and antennas	Ubiquiti 900MHz, Cloverleaf MIMO antenna on rover and dual polarity yagi at base station	Ubiquiti Bullet	Base station antenna was the Ubiquiti AM-2G15-120, rover antenna was the Super Power Supply B00O7ZEK7S, rover and base transceiver was the Ubiquiti Rocket M2	Microhard pDDL2450 could achieve 1km in non-line of sight with 5 dBi omnidirectional antennas. We also backed up comms except the cameras and the TCP link via a RF link with 433 MHz LoRa module.	2.4 GHz and 900 MHz antennas	Ubiquiti M2 Rockets 2.4GHz 802.11n MIMO paired with TP-Link 2408CL omnidirectional antennas	Ubiquiti Rockets M900 and the directional Ubiquiti Loco M900	TP-Link WA 5210 2.4GHz with included directional antenna
Battery System	LGChem18650HE4 Lithium Ion, 80 set up in custom pack, 10 set in parallel with 8 of those sets in series	Custom LiPo modules	1x MaxAmps 7S LIPO Battery	Tattu 6 cell LiPo 22Ah	1x Tattu 6 cell 22Ah Tattu LiPo	Panasonic NCR18650BD 3.7V 3200mAh Li-Ion 4 batteries in series to achieve 14.8V and 6 in parallel to achieve a 19.2Ah	3x Zippy LiPos 7S with a power board we designed	3x 24V LiPo batteries for drive, 2x 12V LiPo batteries for auger/arm
Wired Communication Protocols	I2C, RS232, RoveComm (Custom UDP)	CAN built-into the bananapi with two networks, one for driving wheels, another for the manipulator	CAN bus for interboard, UART for Intel NUC to microcontrollers	I2C for sensors, USB for Raspberry Pi to microcontrollers	CAN for most things, USB for drive motor controller, I2C/SPI for sensors	I2C sensors, USB for cameras, USB serial for Arduinos, UART for GPS, PWM for motor controllers	I2C, UART, Bluetooth (RFCOMM), SPI, PWM, PPM	Serial from the main computer to the various Arduinos
Sensor Fusion	Kalman filtering and custom filtering	robot_ localization	robot_ localization	robot_ localization, custom EKF backup in microcontrollers	robot_ localization	robot_ localization, didn't end up using due to IMU issues	None	None
Team Strengths	Manufacturing capabilities and access to programs that allow us to have many custom components on our rover.	Drive and manipulator controls. Also, I think being just 10 people ups our motivation a lot. Everyone has important work to do	Modularity	Our wireless communication modules. We never lost control or communication to our rover at the competition.	A lot of different experiences from team members because of our coop program.	Tmux for terminal organization, keeping things simple, team dedication, and keeping it fun.	The absolute passion from each and every member of our team as well as our team manage system.	Dedicated team, always ready to learn new things, not shy of challenges. We made great strides in learning ROS in a matter of 3 months.
Improvements for next year	Fix bugs and flaws we found while at URC 2017 and push the boundaries of innovation as we build a new rover.	More field tests of the whole Rover.	Ease of use: easy way to launch and monitor the entire system. Live sensor diagnostics and robust CV.	I really want to add machine learning for finding the tennis ball from further.	Improve our project management.	Clearly labelled wires and pin outs, avoid USB hubs, and a geologist team member.	Secure bigger budget, start earlier, and update our technologies.	More development time, exploit ROS even more, test things more often, more collaboration with other teams.
Source code git	github.com/mst-mrdt	Inverse kinematics only gist.github.com/danielsnider/5181ca50cef0ec8fdea5c11279a9fdbc	https://drive.google.com/open?id=0B1r9QYTd8YNrWXNjNmdtcGlwMjQ	github.com/itu-rover	github.com/uwrobotics	github.com/teamr3/URC	github.com/kammce/RoverCore-S	github.com/Team-Anveshak/rover-control