TELEOPERATED ROBOTIC ARMS WITH OPEN AND CLOSED LOOP CONTROL SYSTEMS

Eli Verbrugge and Brian Dahlman
Mars Rover Design Team
Missouri University of Science and Technology
Rolla, MO, 65409, USA

Advised by: Dr. Kurt Kosbar

ABSTRACT

This paper examines the usage of telemetry for the six degrees of freedom robotic arm designed to compete on a mars rover in the 2019 University Rover Challenge. The arm utilizes three microcontrollers to receive control commands and translates them directly to motor signals for the six brushed DC motors. The usage of the 32-bit microcontrollers facilitates inverse kinematics, an intuitive process that allows commands to be sent as 3D coordinates to the arm, ensuring fine control for arm manipulation. Telemetry is transmitted from the rover to a remote base station over a 900 MHz RF link, using two omnidirectional cloverleaf antennas. Communication between the embedded systems is achieved with the ethernet User Datagram Protocol standard. This ensures seamless transferal of commands from the driver's joystick to the arm, and a stream of telemetry containing motor currents, positional values, and limit switch states - a necessity for the open and closed loop control systems.

INTRODUCTION

Every year the Missouri S&T Mars Rover Design Team (MRDT) designs and builds a rover to compete in four different tasks designed to simulate challenges similar to what would be occurring on a Mars mission as part of the The Mars Society's 2019 University Rover Challenge. Two of these four tasks require the fine control of a robotic arm to manipulate tools, as well as pick up and transport various pieces of equipment [1]. Figure 1 shows the 2019 MRDT Mars Rover in the process of one of these challenges. All of the tasks are performed in Hanksville, Utah, where the Mars-like terrain allows for better task simulation.

The robotic arm is designed to provide six degrees of freedom, and is equipped with DC brushed motors to provide sufficient torque for lifting up to 5 kilograms. The competition requires fine manipulation, such as typing on a keyboard or operating a 4-axis joystick. All of these tasks are performed without a direct line of sight for the operator. The equipment servicing task is performed approximately 250 m away from the base station. Telemetry including video streams and joint angles are a core part of the arm subsystems's functionality, in order to provide the driver with a sense of orientation. The tasks must be performed within specified time constraints, so precise and responsive control is a necessity for a successful design.

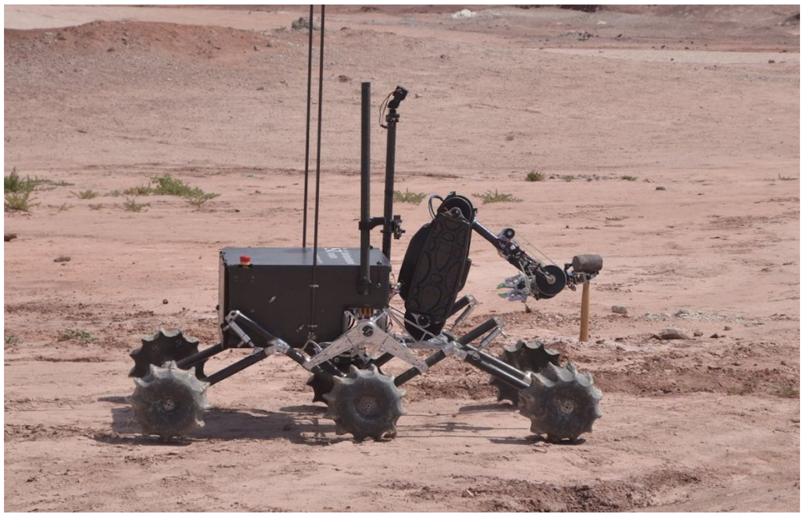


Figure 1: MRDT's 2019 Rover, Valkyrie

The second task is centered around picking up various supplies and traversing terrain to deliver it to a location specified by GPS coordinates and a marker. This poses a less daunting task for the arm subsystem, but does require that the entire system is built to coexist with all other subsystems on the rover.

After many years of competition, it has become clear that we need a stable software platform that does not crash or slow down during tasks, as well as a more powerful control algorithm. This is driven by the increasing strength of the competition, and the need for more robust and fine control. The current version of the arm control framework was built around being modular and allowing the operator to have more control and data at their fingertips.

The software framework was designed to work succinctly with the rest of the communication stack and uses the User Datagram Protocol with a defined packet structure to communicate between all of the embedded systems on the rover as well as over the 900 MHz, 2.4 GHz and 5.8 GHz bands to the remote base station. This allows for the transfer of data between different platforms, and alleviates issues in correctly parsing packets.

This paper focuses on the software and electrical systems on the robotic arm designed to operate the manipulator as well as provide only necessary telemetry to the remote operator. Many of the protocols used in this project are widely used industry standards, and common practices which allow for the systems to be easily expanded upon by others. All of the code and electrical schematics are provided on the MRDT github page [2].

DESIGN

MRDT's robotic arm uses three 32-bit microcontrollers with serial and ethernet connectivity and custom-designed printed circuit boards (PCBs) to control the brushed DC motors, as well as several servos, lasers and a solenoid. Additionally, there are several absolute magnetic shaft encoders and snap-action limit switches whose electrical signals provide information regarding the arm's position and state. Each of the six magnetic shaft encoders has 10-bit resolution, which allows for high accuracy readings, and will retain their positional values even when power is removed [3]. The PWM signals generated from these encoders are proportional to the absolute shaft position and are interpreted by the microcontrollers based on the maximum output. All telemetry is sent off board through the Rover networking infrastructure and to the remote base station.

The software framework is written in C++ and uses several libraries designed by MRDT to generate, and read in, pulse width modulation (PWM) signals using timers provided on microcontrollers, as well as provide timer interrupts for telemetry update purposes.

Due to the limited number of hardware timers on each microcontroller the electrical system uses three separate microcontrollers. One controller parses commands from the remote base station (the drive board), one controls the elbow and shoulder joints (the bicep board), and one controls the wrist joints, as well as gripper and solenoid (the forearm board), the joints in relation to the arm are shown in Figure 2.

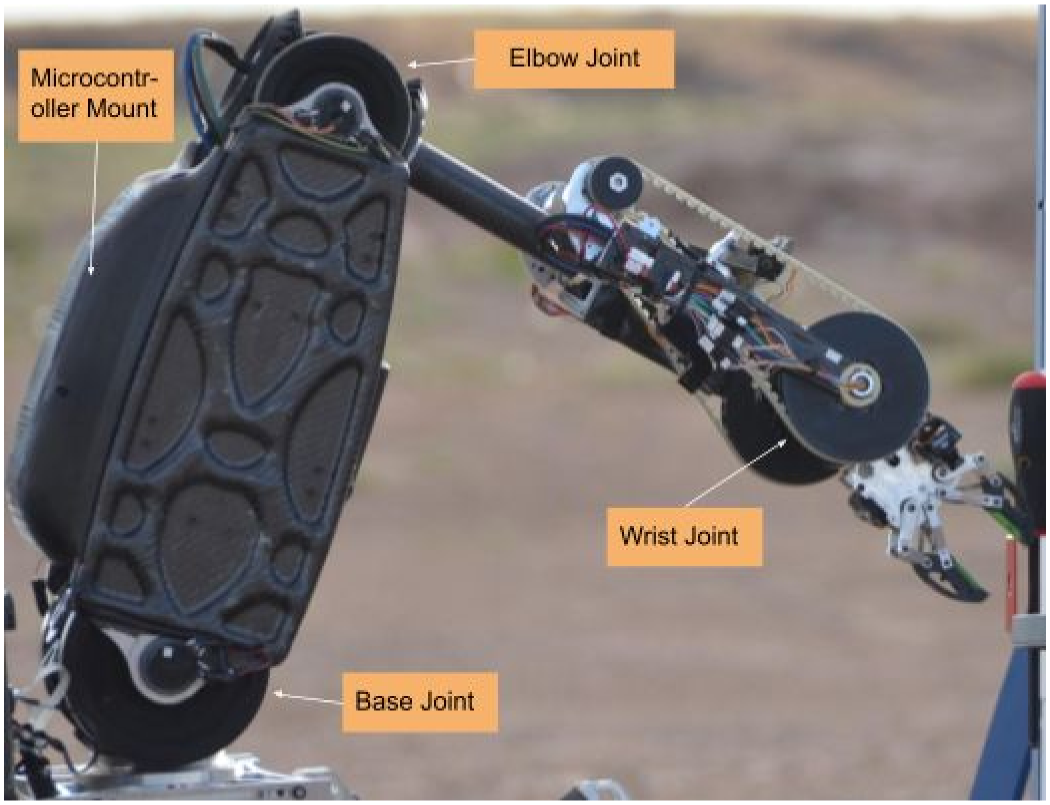


Figure 2: The Arm's 3 Joints

All of the hardware and software is designed to be extensible for any use type of brushed DC motor and encoder one wishes to use to drive a robotic arm with differential joints. Every PCB provides serial connectivity and the microcontrollers have an ethernet port. The software allows for easy parsing of a variety of commands from the remote base station application, and the movement algorithms have several parameters that can be adjusted depending on the load the joint is under and other physical constraints at play.

Software

The software is split into several components. One is a main class for each of the three physical microcontrollers which handles communication between the boards, as well as the execution of commands and telemetry update from the driveboard. Another is the inverse kinematics libraries that provide the functions to calculate joint angles for given coordinates as well as calculating incremental trajectories for real-time movement. These libraries allow for standardization of operation across the several joints, as well as easy sharing of data across the three boards on the arm.

Open loop control is fairly simple as it allows the user to determine exactly how fast and in which direction they want to move a given joint, and requires no calculations outside of a simple formula to facilitate both vertical and horizontal movement in a differential joint. All of the joint axes are mapped to various inputs on an Xbox controller and the only adjustments made are some offsets to counterbalance the weight of the arm dragging itself down.

For each of the joints, we used a tuned PID algorithm to smoothly move our joints to the given angle. This requires a constant stream of commands as well as telemetry between the remote operator and the arm, as the user will be sending destination angles based off of positions from several seconds ago. To ensure as smooth as possible of an operation, and to prevent the arm from trying to move to unexpected positions with user input, a watchdog is employed that will disable all motor operations after four seconds of no communication being received. Encoder values are constantly checked as well, and if a board detects that it has not been reading values from a particular encoder, the system will prevent that joint from being used for closed loop and inverse kinematics.

Control of the utility gripper is achieved via reading in data packets that were sent from base station and parsing them for data that indicates the specific tool being selected. The tools that are not in use are retracted while the desired tool is moved into place.

Inverse Kinematics

Inverse kinematics (IK) is the means of defining a set orientation and position for the arm, and to then calculate the required joint angles to move the arm to the current position. As IK requires knowledge of the current angle of all six joints, it is impossible for each of the separate boards to compute the joint angles necessary for movement without communicating with the others. The third microcontroller, or drive board, is almost solely used for IK calculations. Every update loop the current positions are fed to the driveboard, which will then use this position to calculate the destination angles whenever the remote base station sends a command. To smooth out the path to a given point, trajectories are calculated so that the user can increment a given axis instead of attempt to operate at a much slower pace.

The IK algorithm has been developed and simulated in a numerical computing environment which allows for the algorithms to be independently verified before being implemented and tested within the software framework. This allows for the extensive unit testing and coverage necessary to ascertain the the algorithms perform as expected in a variety of differing initial positions and with a broad range of movement commands.

IK requires a constant stream of info telemetry to prevent offshoot as well as accurate trajectories to the given point. To compensate for various delays in packets or even packet loss, the arm will halt all movement if any given joint is 8 degrees or more away from a target angle so that the arm does not overcompensate in a given axis.

Network

Data is transmitted between the rover and base station over a 900 MHz RFlink, and packets are distributed over the Rover's network switch. All camera feeds are transmitted in the 5.8 GHz band, which requires line-of-sight. The networking infrastructure is scalable, and thanks to the UDP protocol RoveComm, packets can be transmitted to a subset of embedded systems on the rover, allowing for a more direct communication and less network saturation. Each packet sent using RoveComm has a data ID and list of elements, which allows the board to determine what specific task must be done with the set of transmitted data, and standardizes communications across every electrical system on the rover.

To simplify communication with the arm subsystem, the base station communicates directly to the arm drive board alone, which will then send separate commands to the two microcontrollers in charge of driving the six joints, a diagram is provided in Figure 3. This allows for the base station software to be persistent across multiple revisions of hardware. Camera feeds are streamed from two different cameras via a four-channel video encoder, which compresses and streams the camera input of the network to the base station.

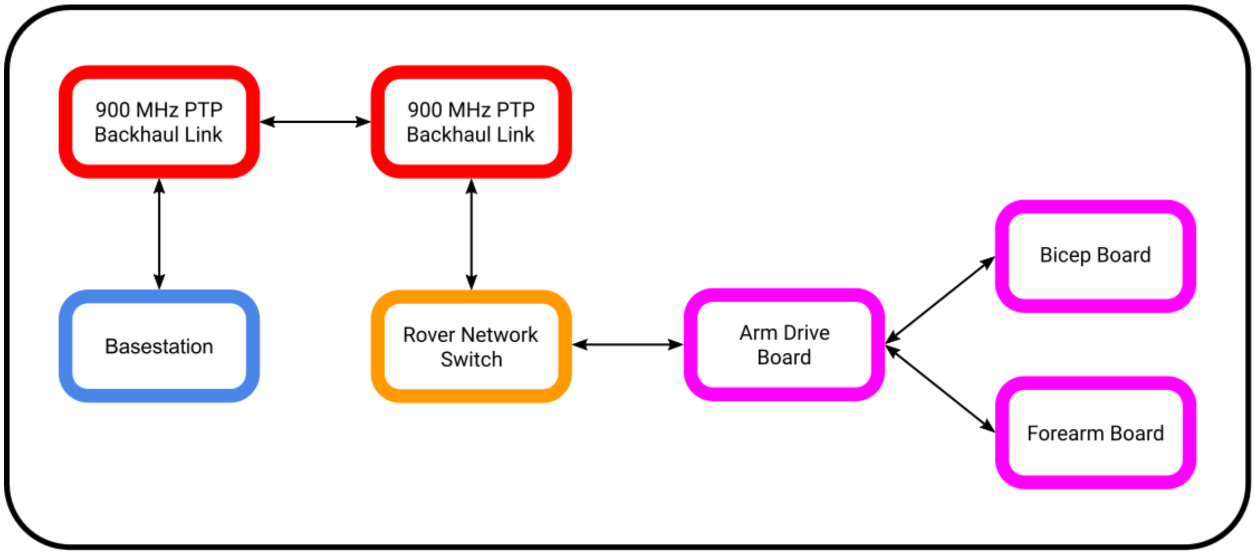
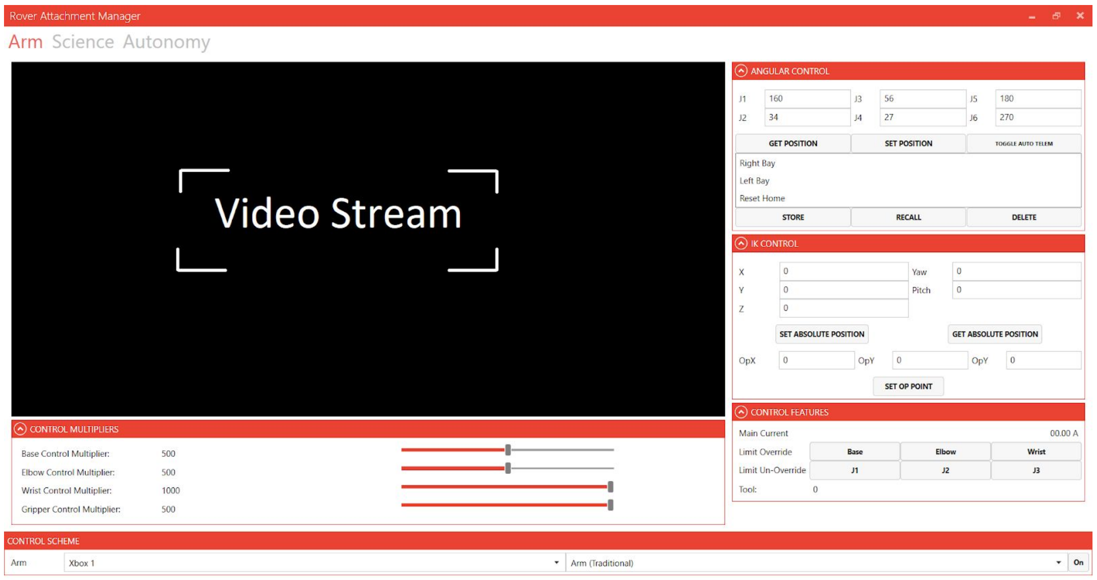


Figure 3: Network Diagram

Base Station

To control the rover remotely, as well as display telemetry and data in as clear as possible to the operator, a C# application is used. The code will display joint angles, motor currents as well as embedded video streams using a VLC plugin, as shown in Figure 4. This allows the operator to have a sense of orientation when controlling the arm. To refrain from congesting the network by constantly sending packets, the base station application will only request arm telemetry when the user interacts with the program. All closed loop control schemes implemented on the arm rely on relative positioning and therefore the base station does not need current positions to drive the arm directly. Control schemes can be modified from within the base station interface and will then directly be communicated to the arm drive board. The base station application is built within a responsive and easy to modify UI using XAML, and using the RoveComm UDP protocol it is simple to add additional network commands to control various robotic arm peripherals such as lasers or solenoids.



J1 160 J3 56 J5 180
J2 34 J4 27 J6 270

X 0 Yaw 0
Y 0 Pitch 0
Z 0

OpX 0 OpY 0 OpY 0

Main Current 00.00 A
Limit Override
Limit Un-Override
Tool: 0

Base Control Multiplier: 500
Elbow Control Multiplier: 500
Wrist Control Multiplier: 1000
Gripper Control Multiplier: 500

Arm Xbox 1 Arm (Traditional)

Figure 4: Basestation Arm View

Printed Circuit Boards

The two armtrain motor controller boards have a 32-bit, 120MHz microcontroller mounted onto them featuring VNH5019A_E motor drivers which are used to drive the arm's six brushed DC motors. The boards are supplied 12 V and have two buck converters that allows for both 5 V and 3.3 V to be used. The motor drivers are powered off of 12 V and read in a pulse-width modulated (PWM) signal from the microcontroller to control the motor corresponding to the chip. The boards are also connected to eight limit switches and four encoders each, which are all powered off of 3.3 V.

The limit switches are connected to digital pins on the microcontroller, which are monitored in code to see if they are tripped. The encoders supply the microcontroller with a PWM signal to indicate the current orientation of the joint to allow for closed loop control of each motor. Each motor is also attached to a push button that can be used to move the motor without needing to rely on basestation to send motor control data packets, which is very useful for debugging.

The robotic arm's inverse kinematics board has a 32-bit, 120MHz microcontroller mounted onto it to run IK on the arm, as well as to control the utility gripper attachment. The utility gripper attachment consists of three positional servo motors, two lasers and a solenoid. The board is supplied 12 V which powers the lasers used for aid in aiming the typing tool. The three servos run off of 5 V which is obtained by passing the 12 V input into a 5 V buck converter and receive a PWM signal from the microcontroller to determine the position of the servo. There is a hex key, phillips head screwdriver and a typing tool attached to their own servo on the utility gripper, pictured in Figure 5. The solenoid which is provided 24 V by means of a boost converter actuates the typing tool which is used to type on a keyboard during one of the competition tasks.

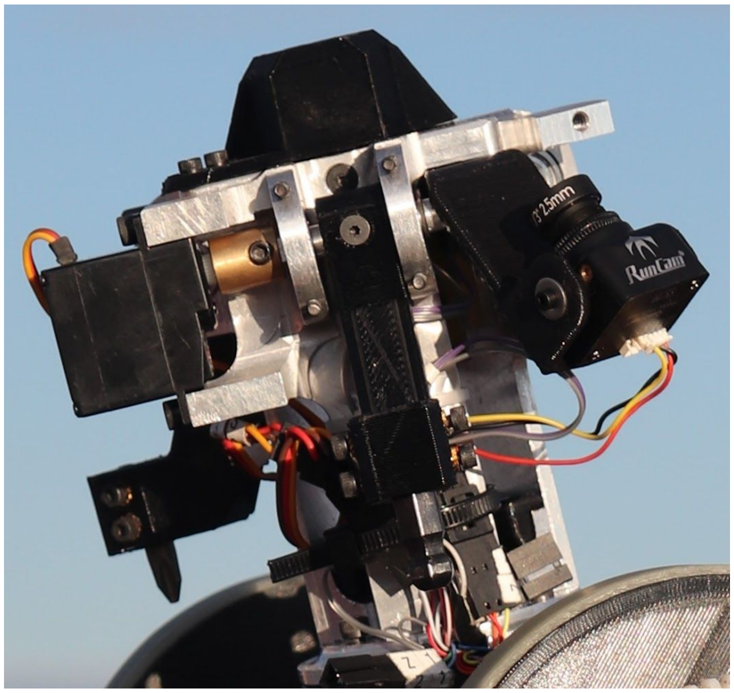


Figure 5: Arm Utility-Gripper attachment

Conclusion

The core software framework and network stack proved to be very robust. After solving a ring buffer issue in the base station application, the arm could be left on for extended periods of time and still send and receive telemetry and commands with no discernible latency. The arm was functional for competition tasks, and all of the joints were fully functional in open loop control.

Progress is being made in developing better methods of PID tuning, allowing the closed loop control to be more smooth and accurate when operating at multiple joints simultaneously. Effort is being put into an application that will intercept arm telemetry and use it to draw movement curves, allowing for more visual feedback when tuning PID parameters.

The ideal outcome of this project as it evolves across future iterations is a modular electrical and software system that can be used for each iteration of MRDT's robotic arm. This will allow for effort to be focused on further research and development of control schemes and movement algorithms. Another goal is to overhaul the low-level components of the software framework, as well as incorporating a hobbyist level brushless motor control to provide an electrical interface which employs a custom open source USB communication protocol to utilize brushless DC motors on robotic arms.

The arm control framework for the teleoperated arm is the foundation of an extensible framework that the Mars Rover Design team hopes will alleviate the brunt of the development process for simple motor control algorithms. The framework has been designed to be modular, and all of the software subsystems can be used independently, depending on the mechanical design of the arm.

This year the team scored 6th out of 36 teams on the Equipment Servicing Task. It is clear that one of the key components to be developed further in the upcoming years for this team's success is to fine tune the control systems and built upon the modular systems to provide the operator with fine manipulation that enhances their ability to perform tasks and does not limit it. The 2019 robotic arm was a successful step in a more configurable system for operation of robotic arms over the rover network.

References

[1] “Requirements & Guidelines.” *University Rover Challenge*, The Mars Society, 2019, urc.marssociety.org/home/requirements-guidelines.
[2] “Missouri S&T Mars Rover Design Team.” *GitHub*, Mars Rover Design Team, github.com/MissouriMRDT.
[3] “MA3 Miniature Absolute Magnetic Shaft Encoder.” *US Digital*®, www.usdigital.com/products/ma3.