

OpenMTR

KUB Capstone project report

Alan Carpenter Matt Clinard Bryan Dingman Nick Hietala Todd Reagan

Spring 2018

**Planning**

OpenMTR utilized multiple project management tools to facilitate the setup of project milestones. The team was divided into specialty assignments:

Matt - Project Management and Development

Bryan - OpenMTR Developer

Nick - OpenMTRDemo Developer

Alan - Website Developer

Todd - Website Content Developer

The team communicated through several online resources including Trello, Slack, and Discord. Trello uses a card system that allows all team members to see what tasks need to be done and who is doing them. Trello updated team members when cards were added, moved, or removed from lists. Slack allowed our team to communicate in real time and allowed for continuous integration with Trello. Discord allowed our team to have meetings when we weren’t all able to meet in one location.

Second, we needed a program to keep any team member from overwriting another member’s code, otherwise known as version control. Git and GitHub were chosen for version control. Git is an open source system designed for software projects big and small. GitHub is a web-based hosting service for version control using Git. It offers all the distributed version control and source code management functionality of Git as well as adding its own features.

Continuous Integration is a software development practice that is based on frequent integration of the code into a shared repository. Each integration is then verified by an automated build. The goal of continuous integration is to identify the problems in the development process earlier and more easily. Regular integration leads to less time spent looking for errors.

We evaluated TravisCI, CircleCI, and Jenkins for continuous integration. We chose TravisCI for a couple of reasons. TravisCI is an automation service that provides support building, deploying and automating our projects. It supports the languages, we chose to use for this project, right out of the box. Also, it’s free to open source projects!

Lastly, we wanted a program that would analyze our code before it was sent out for review. Better Code Hub provides the capability to highlight issues newly introduced during pull requests. This tool helps us keep our code base more in line by checking the code against multiple industry standards and practices.

**Gathering Requirements**

On January 19th, 2018 the team met with representatives from Knoxville Utilities Board (KUB) for our initial meeting. DeWayne Lane, Lead Application System Analyst/Programmer, directed the meeting and gave an overview of the project. He indicated we would be learning an important skill; working as a collaborative team throughout this project. He also revealed the minimum viable product that KUB would require from the team.

KUB is in the process of switching out their utility meters to smart meters. They require an application that will compare the photographs of the non-digital meters with the information reported from their technicians. As it stands, a KUB employee manually reviews these photographs by hand. The application will be required to read in the metadata from the database of photos. It will then iterate through the chosen photos, adhere to a pre-determined confidence interval, and return any photos that do not match the metadata in the database.

Stretch goals, for this project, include the development of an API that processes URL or photo upload, an Azure storage path, a demo of the product, and the application being applied to electric meters.

KUB Application System Analysts/Programmers, Keith Clinard and Zachary Berardo, also joined several of our meetings throughout the semester. They assisted in making sure our work followed the desired path and gave technical guidance. Team members were updated through weekly blog posts and Trello/Slack messages. The KUB team made themselves available through multiple channels of communication to pass along updates or requirements.

**Analysis**

Several challenges were presented with the project. First, the team built the application from the ground up with technologies that were new to the programmers. A second issue presented itself with the optical recognition program. No program is 100% guaranteed to analyze a photo. A third issue involved the overall quality of the photo. The photos presented other issues i.e. dirt in the dial area, glare, and angle of the shot that needed to be addressed.

To address some of these challenges a demo application was built. The demo was used as an internal development tool. This tool allowed the team to make changes to multiple variables at one time. The team was able to change filters via a drop-down box, adjust the blur and angle of the photo, and change the amount of edge detection applied to circles or rectangles.

With a team working on the application and the website, keeping everyone on the same page was critical. Multiple channels of communication were utilized through free communication programs such as Discord. Code was kept free of errors and conflict through GitHub and mandatory code reviews prior to merging code to the master branch.

Once the customer specifications were given to the team, thoughts on how the application would look and act began. The team was required to make a console-based application using the programming language, C#, in a .NET environment. We did not use a database for this application as the data is accessed through a file path. Security protocols were not implemented as no personal identification information was used in the project. The team did not utilize any legacy code or code from another person’s project. Open sources programs were used extensively throughout production.

**Design**

KUB required the application to be able to “read” the numbers from a meter and check them against the metadata from their technicians’ photos. To achieve this goal, the programmers first worked to isolate the area of the dials in the photos. Methods were developed to return rectangle and circle shapes in the photo. Once those shapes were found, the second requirement was to read the number.

Two methods were developed for this issue. Dial meters represented circle shapes. Those circle shapes were then broken down into equal sections that represented the area of a number on a dial. To read the number, the angle of the dial was measured. Removing background “noise” and returning black and white images allowed for dial isolation. Using Hough Circles, all the circles in the image were identified. A gray filter was then applied, and skew was adjusted to straighten the image.

A method was designed to find the center of the dial (circle shape) and the furthest point in the circle. This point, of the dial needle, represented the end of the dial. The needle was located within one of ten quadrants. Based on which quadrant the needle was located, the program will return a number.

The numbers are then compared against the number located in the metadata. If the correct values appear, the value are ignored. If the numbers are a mismatch, a report is return to the technician for manual review or further filter application.

**Implementation**

OpenMTR is a .NET framework console application that utilizes Optical Character recognition to read water and gas meters. Implementation combines the design into source code through coding. We used GitHub to maintain our code repository. We combined all the modules together and detected errors or defects with Better Code Hub. An Agile approach was utilized with Trello as the documentation software.

Other technologies used included OpenCV, OpenCVSharp, NewtonSoft, and JSON. Various frameworks and programming languages were utilized throughout the project. Notepad++, a free source code editor and replacement for Notepad, was used for editing website code. PowerPoint, a presentation program that was released in the 80’s, it is now synonymous with presentations around the world. We used PowerPoint to illustrate our demo application. Atom.io, a hackable web-based text editor from GitHub, was used for version control and code editing. Brackets was designed as a ground up front-end text editor designed for web designers and was used with Markdown editing.

Microsoft Visual Studio 2018, an integrated IDE, from Microsoft that can produce both native and managed code. A .NET framework, in conjunction with the C# programming language, was used to develop our console application and demo program. The demo program can show multiple filters being applied, one at a time, to an image of a meter. The demo shows what the application is doing behind-the-scenes and reduces clutter in the display. The application is now a clean display that returns the essential information of whether the photo matched the metadata. The team will be releasing OpenMTR and will allow for open testing in the market. The program was designed to be open source and is available on GitHub under OpenMTR.