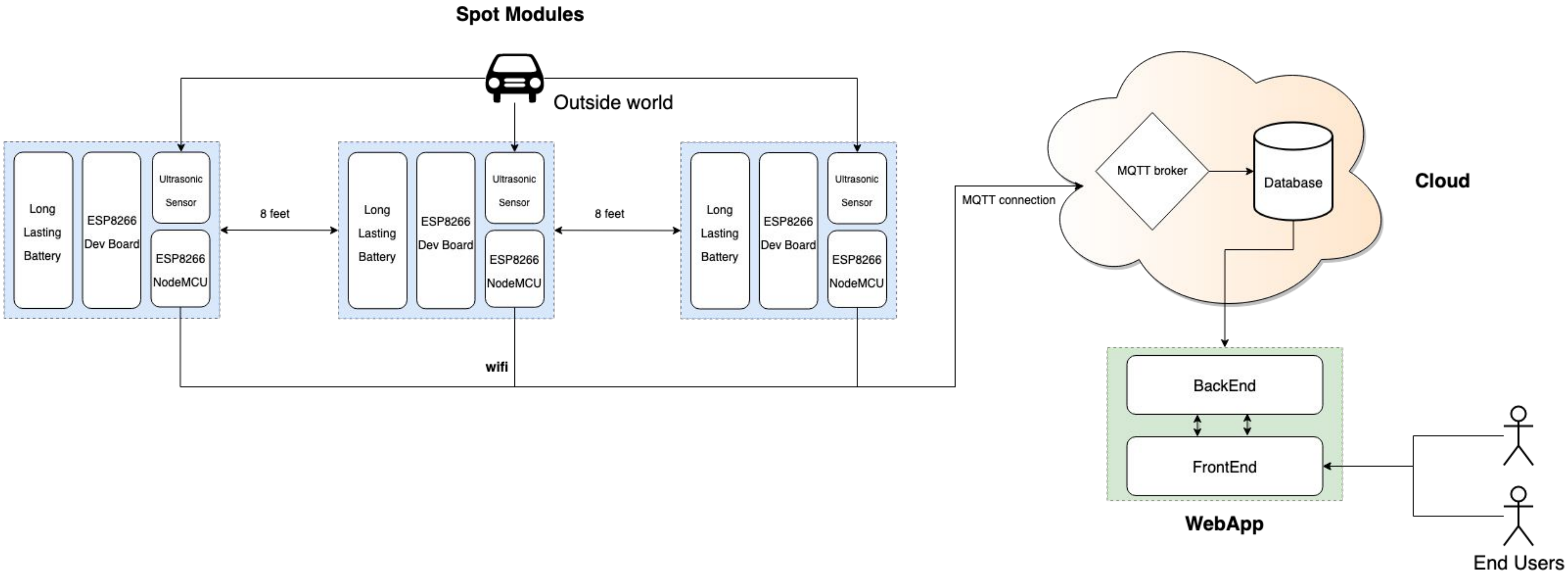
Kerby

Final Presentation

Recap: Kerby's Use Case Requirements

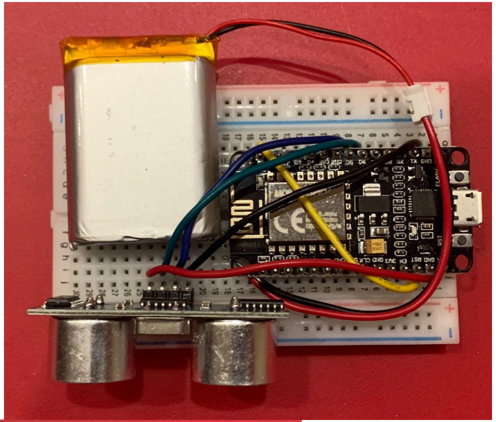
Description	**Motivation**
Car length < 16 ft	Avg. car length = ~14.5ft
Location accuracy within 30ft	30 ft = ~2 cars
Sensors wake up every 5 min	Takes people at least ~3 min to park and leave
Find street parking within <0.5mi to destination	To keep benefit of street parking over garage parking
Cheap spot modules < $50	To enable future scalability

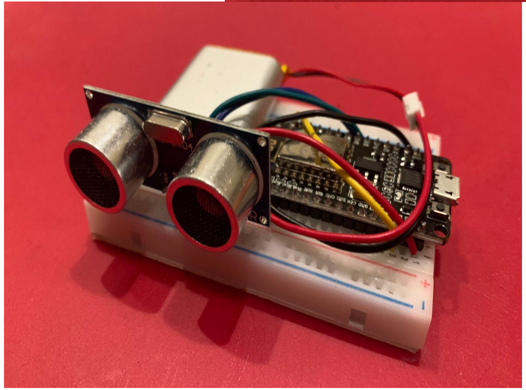
Recap: Kerby's Design Approach



Kerby: Current Solution

- 3 per spot, situated on the curbside
- Fits within 2 in x 4 in x1 in box
- Uses the HC-SR04 ultrasonic sensors
- Uses the ESP8266 module
- LiPo Battery: 3.7V, 1200 mA
    - Added voltage regulator
- Requires Wi-Fi
- ~$20 a piece! $60 per spot
- Limitations:
    - Not weatherproof/compatible with roadside infrastructure yet
    - Mass flashing not available yet
    - Simplified battery replacement process not available yet

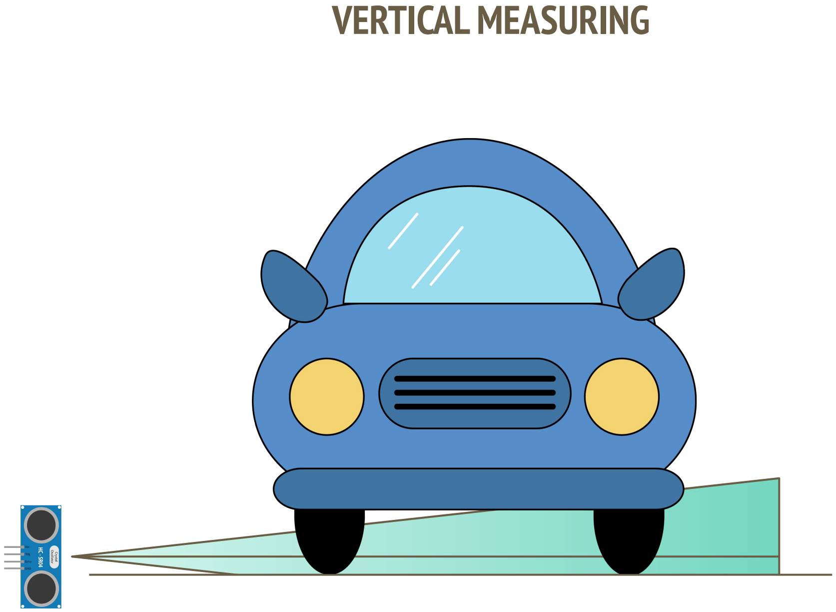


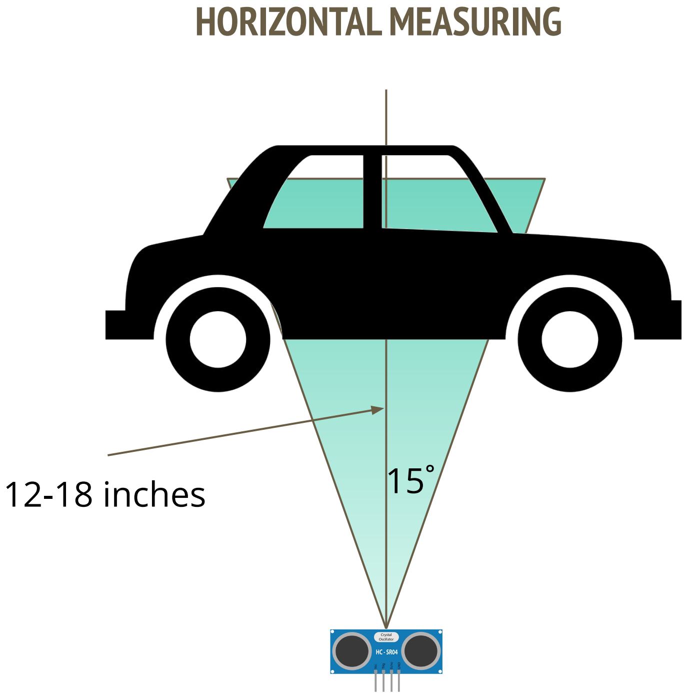


Recap: Testing - Verification & Metrics

Requirement	Measurement	Goal
Find closest street parking to destination	Using google maps api to find distance between given spot and destination	< 0.5 mi
Accurate parking location	Distance between provided location and actual parking spot in real world	< 30ft
Accurate representation of real world	Confusion matrix from testing with large number or users and requests	< 20% False Positive and Negative
Relatively cheap for scalability	Compare to cost of regular parking meters	< $50 per spot module
Easy-to-use web app	User Testing and recording ratings from 1(bad) to 5(great)	> 3.5/5 stars on average
Easy to install	User Testing and recording time	< 5 min on average

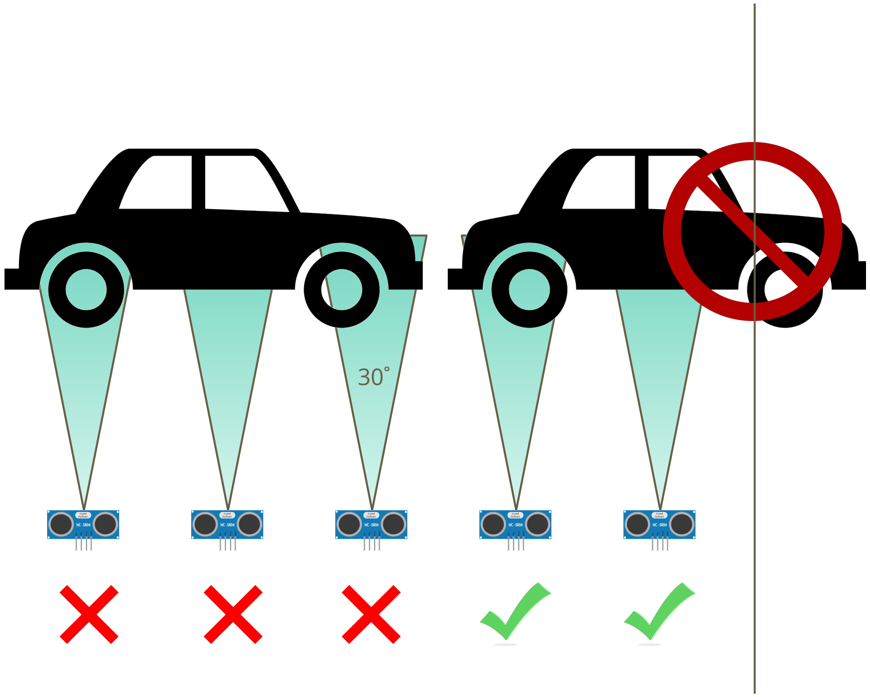
Testing the HC-SR04 Sensors - Range and Reliability





RESULTS

- Positioned the sensors 8 feet apart as planned
- Verified that use-case of <16 ft. cars will be able to use the system reliably, with stipulation of not being extremely high of the ground (>18 inches)

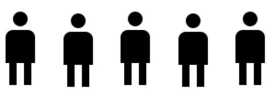


TESTING - DB & Backend core Latency

- Data Upload/Download
  - On avg, using software timing primitives i.e time.future() - time.now()
    - < **1ms** to publish to IoT-Core, <**1ms** to receive message
    - <**1s** to upload to DynamoDB, < **5s** to query and parse DynamoDB results
    - E2E response time per client <<< **3 min** use-case requirement
- Data consistency
  - Issue: ESP8266 Modules with same **1 min** period are flashed to upload out of phase
  - Soln: Query DB data over largest phase difference (**2 periods** instead of **1)** and use most recent
  - Cost: Negligible. < 1ms in query runtime.
- Concurrency control
  - Issue: Enable multi-client processing on sensor readings
  - Soln: Use locking primitives on sensor metadata
  - Cost: Negligible. No deadlock cycles. < 5s increase in request latency

Testing: Web App

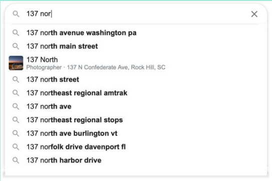
Usability Testing



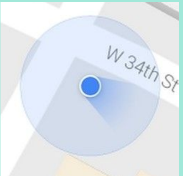
- 5 volunteers with no prior knowledge
- Asked to navigate various pages
- Record feedback after explaining Kerby

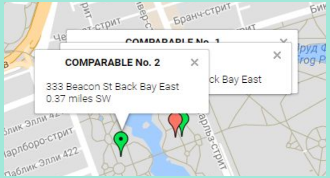
Usability Results

- 4 out of 5 did not like typing full address



- 2 out of 5 thought the route should be more interactive





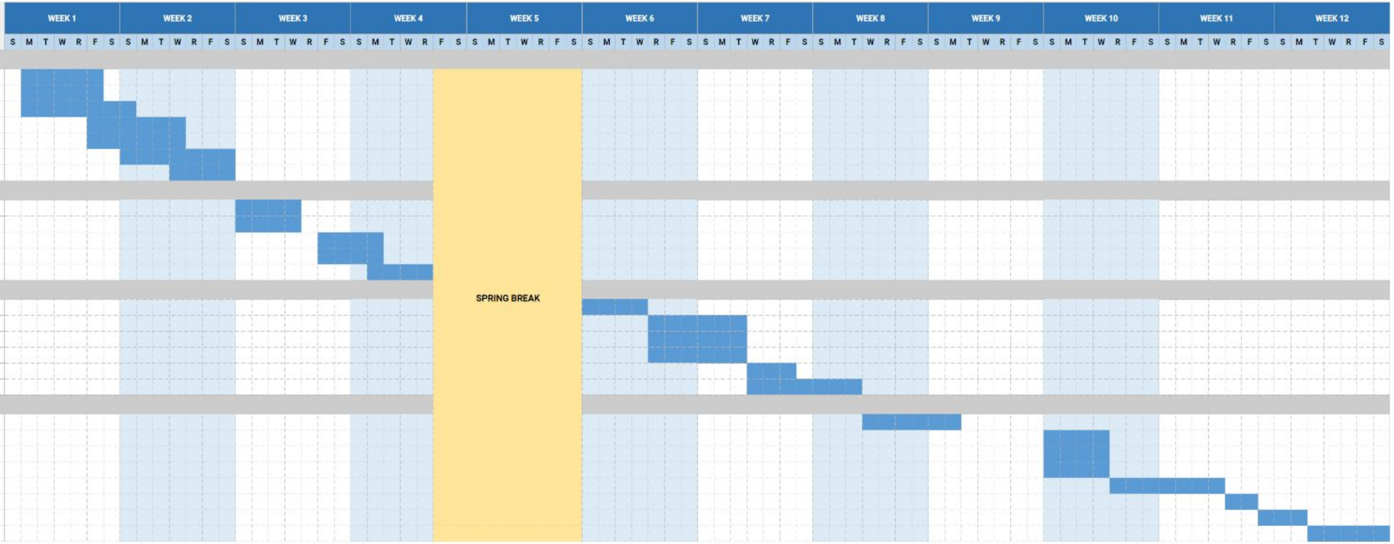
- 4 out of 5 would use Kerby in the future

Project Management Updates

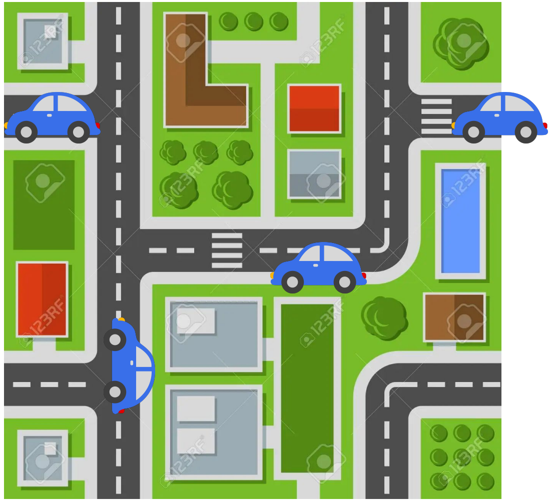
kerby: your curbside parking buddy

TEAM	C1	NAMES	Mrinmayee Mandal, Kanvi Shah, Neville Chima

WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	DURATION
1	**Initial Development**				
1.1	Sensor Research	Minu	2/7/22	2/11/22	5
1.2	IoT Device Research	Kanvi	2/7/22	2/11/22	5
1.3	Database App Research	Neville	2/7/22	2/13/22	7
1.4	Module Set-up (w/o portable power)	Kanvi	2/11/22	2/16/22	6
1.5	Power Supply Research	Minu	2/11/22	2/16/22	6
1.6	Sensor-IoT communication setup	Neville	2/13/22	2/19/22	7
1.7	Design Presentation	All	2/16/22	2/19/22	4
2	**MVP**				
2.1	Module Set-up (w/ portable power)	Kanvi	2/20/22	2/23/22	4
2.2	Web App Back End: simplest algorithm	Minu	2/20/22	2/23/22	4
2.3	Setting up AWS and MQTT broker	Neville	2/25/22	2/28/22	4
2.4	Web App Front End: simplest version	Minu	2/25/22	2/28/22	4
2.5	Design Document	All	2/28/22	3/3/22	4
3	**Improvement**				
3.1	MVP Improvement Research	All	3/13/22	3/16/22	4
3.2	Module set-up (w/ Micropython)	Kanvi	3/17/22	3/22/22	6
3.3	Improve Front End (w/ Google Maps API)	Minu	3/17/22	3/22/22	6
3.4	Improve Communication Protocols	Neville	3/17/22	3/22/22	6
3.5	Integrate Sensors + Database	Kanvi, Neville	3/23/22	3/25/22	3
3.6	Integrate Webapp + Database	Minu, Neville	3/23/22	3/29/22	7
4	**Final**				
4.1	Prepare MVP for interim demo	All	3/30/22	4/4/22	6
4.2	Refine user experience	Minu	4/10/22	4/13/22	4
4.3	Buy new hardware/build casing	Kanvi	4/10/22	4/13/22	4
4.4	Handle multiple queries + modules on database	Neville	4/11/22	4/14/22	4
4.3	Scale Module System	All	4/13/22	4/19/22	7
4.4	Final Field Testing	All	4/20/22	4/21/22	2
4.5	Final Presentation	All	4/22/22	4/24/22	3
4.5	Final Paper	All	4/25/22	5/6/22	12



WHERE IS KERBY GOING FROM HERE?



Public demo will be miniature version of the system implemented on a mock campus.

Remaining features to work on include:

- Viewspots: making it dynamic (i.e. refreshes on its own)
- Portable power/working batteries

Lessons Learned

Through Kerby

- Order hardware early and always order extra!
- Form testing plans and start testing while you build subsystems instead of waiting till end
- Communication can make or break your capstone experience!