Development of an optical communication protocol for control of swarm robots

Benjamin J. Holland[1], Joshua McNeely[2]

***Abstract*—This paper describes the development of an optical based communication algorithm for use in a swarm robot application. A communication protocol is described where packets with 8 data bits are projected to a solar panel on the robot. A simple analog circuit is then used to turn solar panel voltage levels into digital logic levels. A parity bit is sent along with the packet, and a checksum is calculated via a color mapping. The central computer sends an ACK signal upon detecting a valid checksum.**

The scaling of this protocol with the number of swarm members is also discussed and some limitations in it's implementation are considered. Particularly, this algorithm makes use of decentralized computing in order to offset the cost per robot. However, due to limitations on hardware performance, this may not always been the best option. Some workarounds to these issues are also presented.

I. Introduction

TRADITIONALLY, in building a collection of swarm robots, it is imperative that the designer consider cost, communication, and charging in their individual swarm member design. Communication is typically accomplished via radios in the 900MHz-3GHz range. Charging is then accomplished via automated conductive charging, wireless inductive charging, or batteries are manually changed. These separate systems for charging and communication increase the net cost per bot, and do not scale well as the swarm size increases. The number of charging pads and radios increase linearly with the number of bots. An option to decrease the cost and complexity per robot is to combine part functionality.

One option for combining functionality is to combine the data transmission and charging components into a single solar panel. This idea is generally termed "Li-Fi" [1]. This project utilizes a projector/solar panel system for computer to robot communication, and an LED/camera system for bot-to-computer communication. This method allows the data communication and charging to be done in a single unit, reducing cost and complexity of the swarm.

In order to send each robot an individual message, the robot position is tracked using computer vision and unique identifiers called "ArUco Tags" on each robot [2], [3]. ArUco is a library for detecting squared fiducial markers in images. ArUco tags provide a fast, reliable, and unique way to identify the pose of each robot in the frame.

This project is completed as part of an independent study at the Colorado School of Mines.

II. Methods

The main loop in this communication protocol revolves around the detection of the swarm bot pose, sending the message to the solar panel location for each bot, checking the checksum, and then acknowledging the correct message transmission. The process of how the robot interprets messages and how the robot determines the checksum color is described in more detail in section V.

[1] Colorado School of Mines, Mechanical Engineering
[2] Colorado School of Mines, Electrical Engineering

Before any markers can be detected, the camera and projector are calibrated. Camera calibration is done to obtain the fundamental properties of a camera. Because every camera is different (even the same model numbers) this needs to be done for every system. However, once this process is completed, it does not need to be recalculated every time as the camera parameters do not change.

In this project, we use the so-called pinhole model to map 3D points into the image plane using a perspective transformation:

$$s\, m' = A[R|t]M' \tag{1}$$

or

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{33} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

In addition, we consider the radial distortion coefficients, $k_1, \ldots, k_6$ and tangential distortion coefficients $s_1, \ldots, s_4$. The calculation of these coefficients as well as the pinhole model constants are beyond the scope of this paper but are covered in detail in several sources [4]–[6]

Once the camera is modeled, a homography mapping between the projector frame and the camera frame is found. Similarly to the pinhole model, in the homography mapping gives the transform

$$\mathbf{x}_{cam} = H\, \mathbf{x}_{proj} \tag{3}$$

One accurate way to find this mapping is to project a series of known points locations in the projector frame and then find the pixel mapping in the camera frame. To do this a ChArUco board was projected with known corner locations. A sample ChArUco board is shown in figure 1.

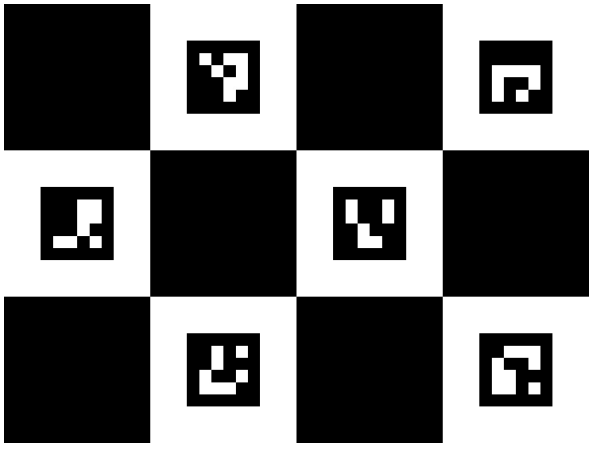

Fig. 1. 4x3 ChArUco board generated using the built in 4x4_50 dictionary.

The camera then takes a picture of the calibration ChArUco board and determines the pixel locations of the each individual corner. A homography mapping is then sought so that the back-projection error

$$\sum_i \left(x_i' - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \cdots + \left(y_i' - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}} \right)^2 \tag{4}$$

is minimized.

An example calibration image from the camera is given in figure 2.

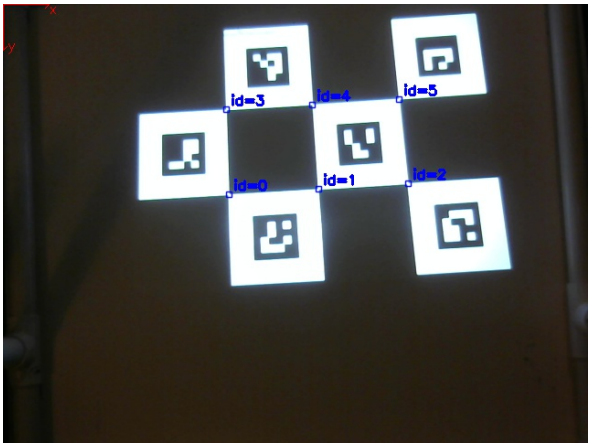

Fig. 2. Deteted corners from a calibration image.

The resulting homography transformation matrix for this image is found to be:

$$H = \begin{bmatrix} 0.35 & -0.02 & 150.7 \\ -0.02 & 0.32 & 34.7 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Once the homography matrix is calculated, it only needs to be recalculated if the relative positions of the projector and camera change.

Now we can accurately use the camera to determine the location of an ArUco tag in the projector frame. This is used to determine the location to project the message bit for a robot's solar panel.

To send a message to the robots, the camera first detects the location of all ArUco tags in the image. Each ArUco tag point is then used to determine the relative location of the solar panel in the tag's frame (assuming the tag is fixed to the same location on each robot). A white or black polygon is then drawn in the offset area depending on whether the message bit is 1 or 0. This detection, offset, and projection process is completed for each bit in the message.

III. Base Station Hardware

The hardware for this project is separated into two categories: (1) base station and (2) Swarmbot member.

The goal of the base station is to decentralize the computing hardware and decrease the cost per part. In addition, it provides a centralized location to do command and control of swarms. Often, this includes position feedback and correction, data transmission, and swarm maintenance. For this project, the following items were used for the base station command, detection, and projection.

- Projector - NECV300X [7]
  - 60Hz Operation
  - 12.5 sqft Projection Area
  - mounted 4ft above arena
- Logitech C270 Web Cam [8]
  - 640x480 Resolution
  - 60° field of view
  - 30 FPS max
- Raspberry Pi Model 3
  - Quad Core 1.2 GHz
  - 1 GB RAM
  - Ubuntu Mate (Linux 4.4.38)

IV. Robot Hardware

The individual swarm bots were built to minimize cost so the swarm scaled well with size. The main hardware on each bot is described below.

- Solar panel - Seeed Technologies 1W Solar Panel [9]
  - Capable of detecting incident light intensity changes
  - Area = 40cm^2
  - Quiescent Voltage level $V_{oc,inside} = 3.9V$
  - Signal Voltage level $\Delta V_{signal} = .2V$
- LED - WS2812b [10]
  - capable of encoding messages via color
  - Power out = .150W max
- Adafruit Trinket - 8MHz ATTiny 85 [11]
  - Main processor on board robot
  - RAM/Flash amount
  - 5 IO Pins
- 9g Micro Servo modified for continuous rotation [12]
  - Differential drive
  - All driving circuity is self-contained, require sonly a PWM signal to drive

A. Chassis

The chassis is cut from 1/4" acrylic. The wheel brackets are cut from within the chassis, allowing for minimal material waste in robot production. The chassis also features numerous holes to allow bolting of the various components onto, including:

- PCB
- Solar Panel
- Ball caster
- Battery
- LED
- Servos
- Accessories specific to a bot type

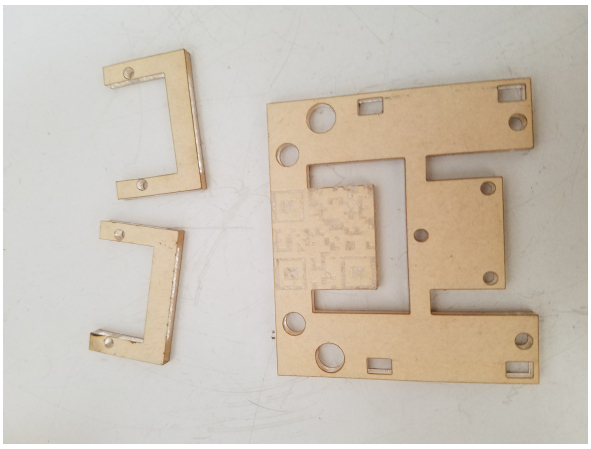

Fig. 3. A swarmbot chassis

B. Circuitry

Consistent with the hardware component list above, each bot features several electrical subsystems working in unison.

1) Filter: To send data to a specific robot, an overhead projector displays a white or dark circle centered on that bot's solar panel to create a voltage difference on the panels output. This change in intensity is small, and the digital inputs of the microcontroller are not able to discern the small ΔV of the unconditioned panel voltage. In order for the robot to interpret this signal, an inline filter is used to ensure a logic level output to the microcontroller.

The LT1013 is used to build a comparator circuit. To ensure the filter output is independent of the light intensity of the surroundings, a simple RC high pass filter is included before the signal goes to the LT1013. When a white square is "written" to the solar panel, the voltage on its output increases. This change propagates through to the comparator, and if the filtered signal goes above the reference value, the amplifier saturates the output at V+. The other scenario (signal is smaller than reference) sets the output at V-. For our purpose, V+ is set at 3V and V- is ground, allowing the digital inputs of the microcontroller to read the message the projector is sending.

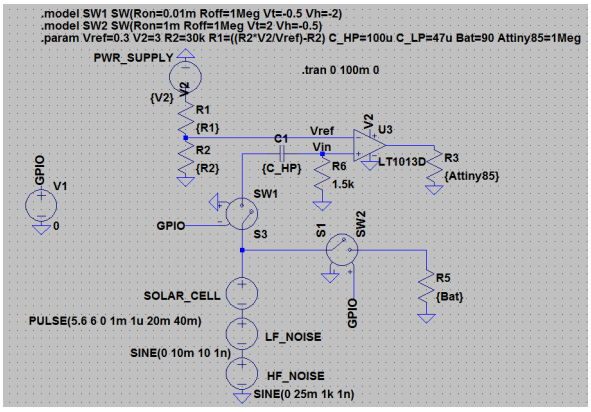

Fig. 4. Schematics of the solar panel circuit for filtering and charging

2) PCB: The circuitry for each Swarmbot member is made compact via a printed circuit board designed by the group. Using the software Fritzing [13], a PCB was designed allowing for all components to fit within a small footprint.

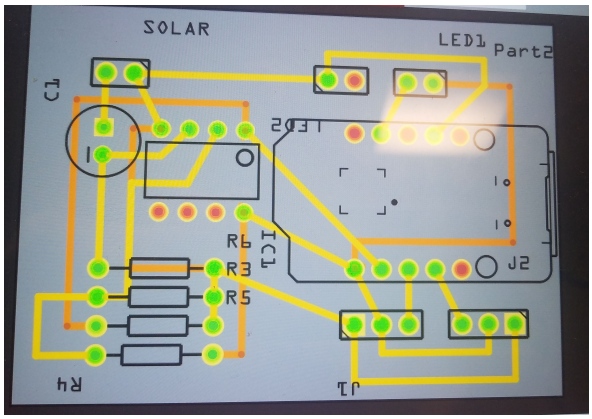

Fig. 5. The design within Fritzing

Once designed and verified using Fritzing's tools, the PCB was milled using Mines' T-Tech mill.

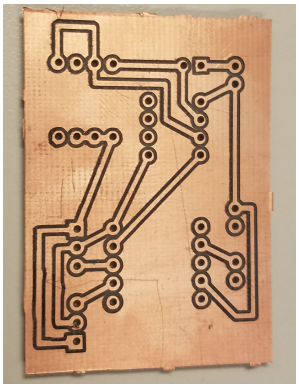

Fig. 6. The bare PCB fresh from the mill

Once milled, the components were installed on the PCB.

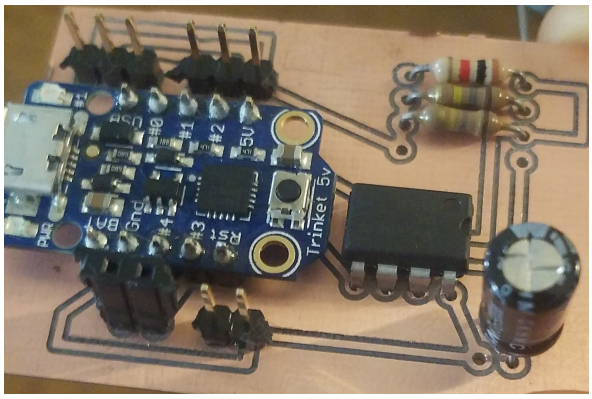

Fig. 7. PCB with components installed

V. Communication Protocol

The implementation of Li-Fi to swarm robots comprises two distinct members: the base station and individual swarmbot members. The hardware present requires two different communication methods to achieve two-way communication between a bot and the base station. Each system is described below.

A. Robot to computer

Each bot's onboard LED is used as the transmitter, and the base station's overhead webcam acts as the receiver for this subsystem.

As the bot's LED is capable of 8-bit color resolution in each R, G, and B channel, information is sent encoded in the color of the LED.

With the correct LED for the robot known, it is necessary to describe the two different regimes of communication, non-communication mode and base station control mode. Each distinct regime is described below.

1) Non-communication Mode: In this first case, when robots are not being actively controlled, they transmit their status periodically to the base station. Every possible robot status maps to a color according to the following:

- Red: Low battery
- Green: Idle
- Blue: Lost
- Yellow: Doing Action
- Orange: Charging
- Purple: Asleep

The LED pulses on for 1 second every 5 seconds to keep power use low. The overhead camera is able to identify the color of this LED to determine a robot's status and thus make actions when the swarm requires intervention.

2) Base station control Mode: In the case when the base station talks to the robot, the LED is used to transmit a "color"" checksum to the base station to confirm the transmission was received and interpreted correctly. As above where each possible status maps to a color, each possible message maps to a color according to the chart below.

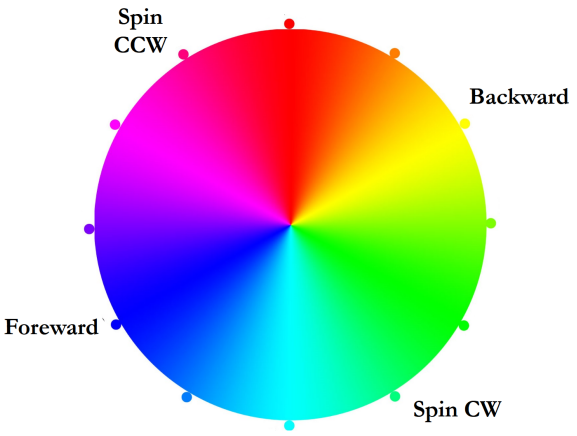

Fig. 8. Color checksum mappings

Arc paths are simply linear combinations of spins and straight lines, so every type of message is representable as a color. When a message is sent, the computer computes the expected color and cross references it against the color reported by the robot. If they match, the base station sends a final ACK bit.

B. Computer to Robot Communication

Once an individual robot's pose is known, the overhead projector can send data only to the specific location corresponding to that robot's solar panel. This not only allows each robot to receive individualized commands, it also allows all communication to be done in parallel.

For communication from base station to bot, the overhead projector is used as the transmitter and the solar panel/filter is used as the receiver. Bits are transmitted by writing circles of either LOW or HIGH intensity indicating binary 0's and 1's respectively. The varying intensity of the squares effects the voltage on the solar panel which is filtered and interpreted by the bot's microcontroller.

This allows the projector to send binary HIGH and LOW signals to the robot. These bits are composed into packets of the following form

Frame (12 bits)					
Header (2 bits)	Data (8 bits)			Parity (1 bits)	ACK (1 bit)
Start Bits 1 0	LWheel Speed Data[0:2]	RWheel Speed Data[3:5]	Duration Data[6:10]	Parity Bit Even Parity	ACK Bit

Fig. 9. The bits comprising a single packet. 8 data bits are in sent in each transmission using a 14 bit frame.

- Header (2 bit): the bits to begin the message
- LWheel: set the left wheel speed from -7 to 7
- RWheel: set the right wheel speed from -7 to 7
- Duration: set the duration of the action from 100 to 800ms
- Parity: a bit to ensure message integrity
- ACK: sent after the rest of the bits are received and processed, described below in the color checksum

Each packet is comprised of 2 header start bits, 8 data bits, 1 parity bit, and a final ACK pit. In the idle state, the square being projected onto the bot's panel is dark, indicating a 0. To begin communication, the projector writes a light square followed by a dark square, indicating a "10". The bot detects the initial 1 and waits 1 period to ensure the following reading is 0.

Upon reception of a "10", the bot then reads an additional 9 data bits. The first 8 comprise the data and the final bit is the parity bit. The bot then computes the parity of the first 8 bits received. This computed parity is compared with the received parity bit to check message integrity. The parity bit can fail in the case of burst errors, so even if the parity check is successful an additional layer of error checking in the form of the "color checksum" described above is used to minimize corrupt transmissions.

Once the color checksum is written, the bot begins waiting for an ACK bit while the overhead camera on the base station reads the updated value of the LED. The base station performs the same color mapping on the transmitted message and compares this to the color detected on the bot's LED. If the colors match, the computer sends a single ACK bit indicating the robot is clear to perform its action. This ensures that if a corrupted message is received and a parity check is successful it will fail to cause erratic behavior in the bot because it will not pass the color checksum.

VI. Protocol Analysis

This setup is technically capable of full duplex communication, since the projector can be writing to the robot while the robot is writing to its LED. However, as implemented it is half-duplex since the robot can only read or write at one time. Additionally, since the clocks of the projector and the robot are not synchronized, this amounts to asynchronous serial communication.

A. Computer to robot

This subsystem can be analyzed by looking at the speeds and capabilities of the individual hardware components used.

- Central computer: 4Ghz Quad-core processor
- Projector: 60Hz refresh rate
- Solar Panel/Filter (assuming acting as a photodiode): ≈2kHz
- Bot microcontroller: 8 MHz

1) Bit Rate: As can be seen from the rates, the bottleneck in this system is from the project's refresh rate. Thus, the theoretical maximum bit rate of the computer to robot communication system is constrained to how fast the projector can update,

$$R_{max} = 60bps$$

This is the total data rate, include the header and parity bits. The rate of data transmission is

$$R_{data,max} = \frac{N_{bits,data}}{N_{bits,packet}} R_{max} = 40bps$$

2) Entire swarm rate: The above analysis describes half-duplex communication with a single bot. However, due to the high resolution of both the projector and camera combo, they allow parallel communication with numerous bots simultaneously. As each bot is capable of communicating independently with the computer, the data rate for the entire swarm scales linearly with the swarm size

$$R_{b,swarm} = N_{bots} * Rb, bot$$

This data rate is the theoretical maximum. In reality, it has some practical limitations, mainly the central computer's processing power and the resolution of the camera and the projector. Luckily, the pixel size of the projector is much smaller than the footprint of each bot's solar panel, so the projector resolution causes no bottleneck. Similarly, the camera has high enough resolution to discern each bot's LED clearly. Issues would arise were the size of individual swarm members to become roughly 2 orders of magnitude smaller.

VII. Results

Using the calibration and detection process described in the methods, we were able to accurately detect and project messages onto a solar panel location. A sample of input, detection, and output images are shown below.

Benchmarking was also performed on various parts of the detection and projection loop. This was done to capture how the method scaled with the number of markers and which aspect were unaffected. The average time and standard deviation for up to eight markers is shown below in figures 13, 14, 15, and 16.

VIII. Discussion

One of the largest issues faced here is the hardware limitation on the base station. Currently the largest bottleneck is in the Raspberry Pi's ability to detect, process, and display markers on the projector. During testing, the fastest consistent clock rate that we could achieve was 4 frames per second (up to 8 markers), 90% slower than the theoretical maximum of 40 bits/second.

Looking at the benchmarking results show that the two longest operations were the marker detection and homography application. These processes are very processor intensive and could be sped up

Fig. 10. Source input image

Fig. 11. Source image with detected markers shown in blue and rejected marker candidates shown in red.

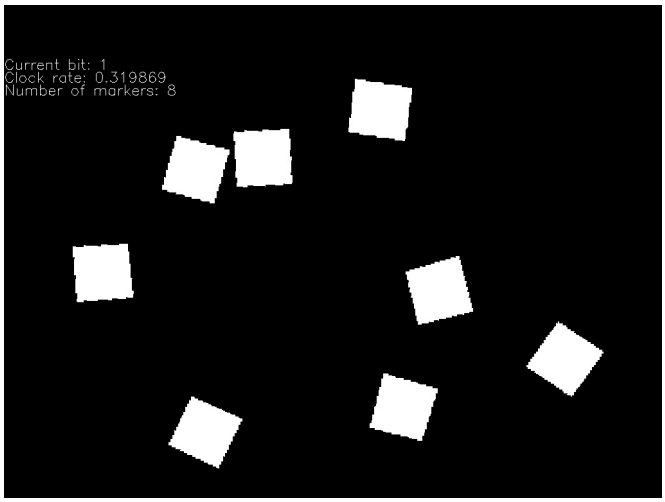

Fig. 12. Projector output image with source file removed, homography applied, and a data bit projected offset from the marker.

TABLE I
BENCHMARK PERFORMANCE TIMES FOR VARIOUS PARTS OF THE DETECTION, PROCESSING, AND PROJECTION LOOP. MEAN TIME IS SHOWN IN SECOND AS WELL AS THE STANDARD DEVIATION FROM THE MEAN.

# Markers	Full Message	Detection	Homography	Pose Estimation
1	$2.58 \pm .040$	$.061 \pm .004$	$.073 \pm .005$	$.003 \pm .001$
2	$2.59 \pm .026$	$.063 \pm .004$	$.073 \pm .005$	$.004 \pm .001$
3	$2.59 \pm .029$	$.064 \pm .006$	$.073 \pm .005$	$.007 \pm .001$
4	$2.60 \pm .028$	$.065 \pm .005$	$.073 \pm .005$	$.009 \pm .001$
5	$2.60 \pm .033$	$.067 \pm .006$	$.073 \pm .005$	$.010 \pm .001$
6	$2.60 \pm .029$	$.067 \pm .005$	$.073 \pm .005$	$.012 \pm .001$
7	$2.61 \pm .036$	$.067 \pm .004$	$.073 \pm .004$	$.015 \pm .002$
8	$2.78 \pm .157$	$.066 \pm .006$	$.073 \pm .004$	$.020 \pm .004$

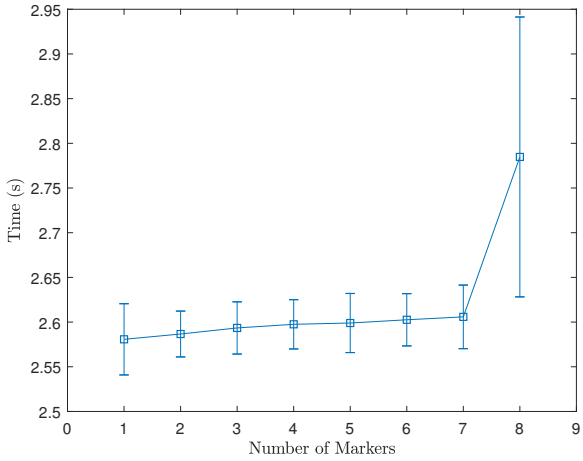

Fig. 13. Average time to send entire message for up to 8 markers. Error bars represent standard deviation from the mean.

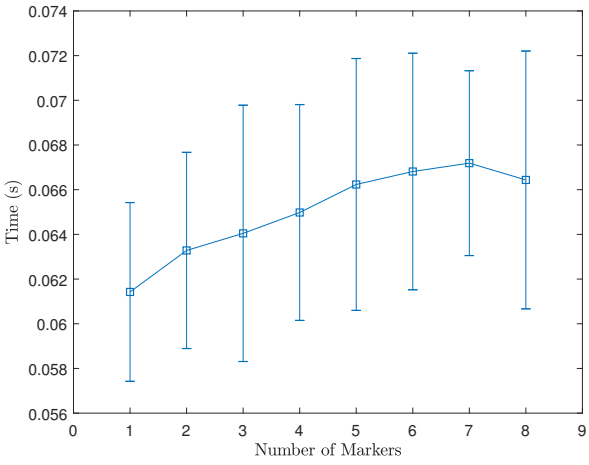

Fig. 14. Average time to complete marker detection for up to 8 markers. Error bars represent standard deviation from the mean.

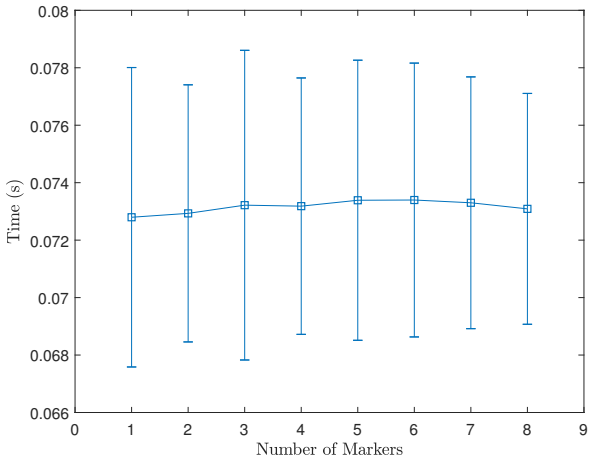

Fig. 15. Average time to apply homography transformation for up to 8 markers. Error bars represent standard deviation from the mean.

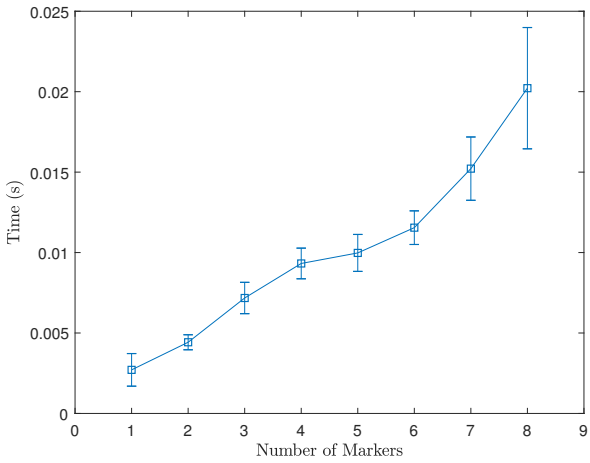

Fig. 16. Average time to estimate pose of all markers for up to 8 markers. Error bars represent standard deviation from the mean.

using parallel programming or through faster matrix multiplication algorithms. One option could be to use sparse matrix operations due to a majority of the projection image having no value. Additionally, more aggressive image refinement before marker detection would allow for fewer detection iterations before all markers are detected.

The Raspberry pi was selected for this project because of it's low cost. However, it proved to be ineffective as a swarm controller. Many of the operations performed would benefit from increased CPU clock speed and a dedicated graphics card. Because the Pi's graphics capabilities are integrated with the processor, processing images and projecting data bits have to be done one after the other instead of in parallel. Because of these limitations, we were unable to achieve any respectable clock speed.

Recent development's in the ArUco library claim speed improvements of at least one order of magnitude. This would drastically increase the marker detection speed and also provides better support for detection of markers in video streams.

Despite these short comings, this method still has several advantages over traditional methods of communication. Namely, adding robots to the swarm only increases the amount of time spent calculating pose estimation. All other main components do

not scale with the number or robots. This allow robust parallel communication with all of the bots in the swarm. However, in order to outperform a serial communication protocol operating at 9600 bps, there need to be a minimum of 160 bots in the swarm (assuming theoretical best performance of this method).

One of the other benefits of this method is that is that the cost scales well with the number of bots. When introducing a bot to the swarm, one just needs to verify that it has a unique tag and the base station handles to rest.

Additional challenges with this method arise when considering projector and camera capabilities. The projector must have enough intensity to trigger a state change in the solar panel. This can become particularly challenging when the robots are in areas of bright light or rapidly changing lighting conditions. In addition, the projector is resolution limited and can only project in discrete pixel location.

The camera suffers from related issues. Again, the images are limited by the resolution of the camera. If they tags become too small to be resolved in the detection algorithm, communication with the robot may be lost. In addition varying lighting conditions (particularly in low light) can cause the camera to miss markers.

A. *Task-specific bots*

As the majority of the effort has been spent in setting up the swarm, little though has gone into what the bots do once they function. As is, the bots can simply move around and produce colors, but they feature little ability to manipulate their environments. Keeping with the premise that any new system must be low cost and able to be integrated with the current platform, the group prototyped several different bot types. Below are pictures with short captions describing their functions.

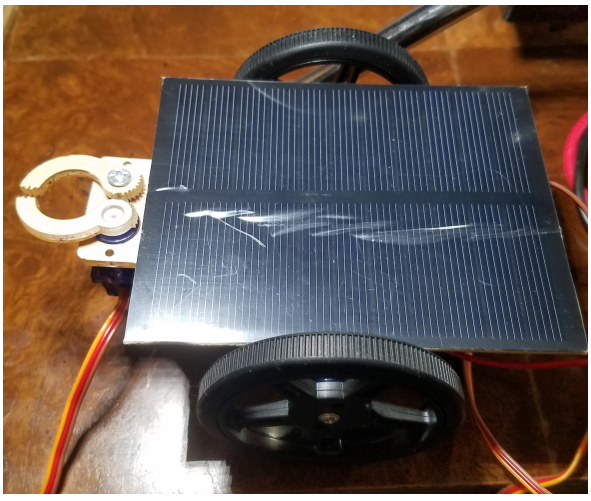

Fig. 17. A pincher bot able to pick up foam blocks.

Fig. 18. A crane bot able to pick up and drop ferrous objects (via an electromagnet).

IX. Conclusion

A method of two-way asynchronous communication is described utilizing a projector, solar panel, RGB LED, and camera. Realized parameters are discussed from a hardware implementation in the form of swarm robots and a base station. A communication protocol is described, allowing half-duplex communication at bit rates of roughly 10bps for each bot.

The methods used in this project proved to be quite successful in terms of a proof of concept for a swarm robot system. Overall, this project provided successful demonstration of

1) Swarm tracking using ArUco tags
2) Data transmission using Solar Panels
3) Wireless charging for swarms
4) Cost reduction of robots

Though a full system model was not created, each subsystem was able to individually complete the appropriate actions. Future work to be done on this project involves optimizing the individual components in order to maximize the potential of the system and integrating the components together into full working model. Opportunities for improvement include:

1) Faster charging (higher power leds)
2) Lower cost robot (custom PCB)
3) Unique bots for specific tasks

Finally, though this method provides some promising ways to scale the swarm size, it also has some challenging implementation constraints in regards to the base station, projector, and camera. An an ideal scenario, this would be implemented using a power processing computer, high power and high resolution projection, and high resolution camera. Fortunately, these areas of technology are rapidly improving and it may not be long before these constraints become negligible. Additionally, one could move away from consumer hardware and the visible spectrum to achieve better results with more accurate detectors/transmitters.

The code used to run this project can be found at https://github.com/behollan/SwarmBot.

References

[1] Harold Haas. Wireless data from every light bulb. https://www.ted.com/talks/harald_haas_wireless_data_from_every_light_bulb, 2011.
[2] Open computer vision. http://opencv.org/about.html. Web. 14 March 2017.
[3] S. Garrido-Jurado, R. Mu noz Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292, 2014.
[4] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.

[5] Frédéric Devernay and Olivier Faugeras. Straight lines have to be straight. *Machine Vision and Applications*, 13(1):14–24, Aug 2001.
[6] R. I. Hartley. Projective reconstruction and invariants from multiple images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(10):1036–1041, Oct 1994.
[7] NEC. Np-v300x 3000-lumen high-brightness mobile projector. http://www.necdisplay.com/p/multimedia-projectors/np-v300x?type=support, 2017.
[8] Logitech. Logitech c270 webcam specifications. http://support.logitech.com/en_us/product/hd-webcam-c270/specs, 2017.
[9] Seeed Technology Co. 1w solar panel. https://www.digikey.com/products/en?mpart=313070005&v=1597, 2017.
[10] Sparkfun. Led - smd rgb (ws2812). https://www.sparkfun.com/products/11821, 2017.
[11] Adafruit. Adafruit trinket 5v. https://www.adafruit.com/product/1501, 2017.
[12] Adafruit. Continuous rotation micro servo. https://www.adafruit.com/product/2442, 2017.
[13] Fritzing. Fritzing. http://fritzing.org/home/, 2017.