Fig. 1. Picture of the Saguaro INN. The solar panel that provides power to the unit and the sensor attached to the stalk are visible.

The Saguaro INN is designed to be placed in remote locations and communicate via cellular data networks. We use either a calibrated pyranometer (Apogee SP-212) or a cheap silicon photodiode (Osram BPW34) to monitor the irradiance at the location of the sensor. We use a trans-impedance amplifier to convert the current from the photodiode to a measurable voltage. Voltage data from the sensor is read every second from a program running on an Olimex iMX233-OLinuXino-MICRO via an analog to digital converter. This data is sent every minute via a cellular modem (Multi-Tech MTSMC-H5) to our central database. This hardware is co-located on a custom PCB along with a DC-DC switching power supply, a board temperature monitor, and a battery voltage monitor. A 10W solar panel, 6Ah, 12V lead acid battery, and solar charge controller provide power to the hardware that consumes less than 1W of power on average. A picture of the Saguaro INN is shown in Fig. 1.

The Prickly Pear and Yucca INNs both send data over the internet using a Raspberry Pi, but use different sensors. The Prickly Pear INN communicates over Ethernet to a rooftop PV system's monitoring hardware (e.g. SMA Sunny WebBox) to use PV power as a proxy for irradiance. The Yucca INN uses a pyranometer or photodiode, like the Saguaro INN, that is read via an Arduino FIO placed in the sun and transferred via XBee radio to the central unit.

All three INNs use the Arch Linux operating system with custom kernels. Most programs are written in Python. Data is time-stamped on each INN, and the INN clock is synced via NTP. We currently use SFTP to transfer the data from the sensor to our central database every minute. In the future we will use the messaging service ZeroMQ to transfer data with lower latency. We can update the software remotely using Fabric, and we use SSH to remotely log-in to a sensor if needed. We also have scripts on the Saguaro INN that monitor the cellular data connection, the board temperature, and the battery voltage. The INNs only send data during the day (as calculated by ephemeris code for each day) to save power and network bandwidth.

B. Central Database

As soon as data is sent to our central server via SFTP or ZeroMQ, a script loads the raw data into a MySQL database. The data is identified by a sensor ID number, epoch time stamp, and measurement. We also keep a MySQL table to store metadata for each sensor including location, sensor type, etc., and a table to store battery charge levels and temperatures for Saguaro INNs.

C. Network Used in this Study

A map of the 19 sensors used in this focused study is shown in Fig. 2. Most of the sensors are Saguaro INNs, although some are 5-minute data from rooftop PV systems, and 2-second power data from utility-scale installations. Forecasts were analyzed for locations at the UASTP, shown in Fig. 2b. We use a higher density of sensors to the southwest of the UASTP because the primary wind direction is from the southwest.

III. Sensor Network Forecasts

Here we describe how we generate our network forecasts, which is similar to our previous method described in [6]. First, we generate clear-sky expectations for each sensor using data from clear days. Operationally, these are generated weekly and checked visually. This data driven approach captures shading due to obstacles, orientation, and other system specific parameters. At each specified time step t (every 1 minute in this study), we calculate the clearness index for each sensor n as

$$K_n(t) = \frac{g_n(t)}{g_{n,clear}(t)} \tag{1}$$

where $g_n(t)$ is the measured data at time t and $g_{n,clear}(t)$ is the clear-sky expectation at time t. In this study, the measured data is the average of the data collected over the previous one minute. Once the clearness is calculated for each sensor, we use bi-variate interpolation to make an interpolated clearness map similar to Fig. 3. We set the boundary of this map using the average clearness of all sensors for the previous minute. We can also use satellite images or numerical weather models to set this boundary. Then, we forecast the clearness for a sensor or arbitrary location by propagating this clearness map using an assumed cloud motion vector. There are several ways we can estimate this cloud motion vectors, and we describe some methods we will explore in future work in Section V. Here, we use the hourly outputs of a custom, high-resolution Weather