Estimating Outdoor Temperature from CPU Temperature for IoT Applications in Agriculture

Chandra Krintz, Rich Wolski, Nevena Golubovic, and Fatih Bakir
Computer Science Department
University of California, Santa Barbara

ABSTRACT

In the paper, we investigate using CPU temperature from small, low cost, single-board computers to predict outdoor temperature in IoT-based precision agricultural settings. Temperature is a key metric in these settings that is used to inform and actuate farm operations such as irrigation scheduling, frost damage mitigation, and greenhouse management. Using cheap single-board computers as temperature sensors can drive down the cost of sensing in these applications and make it possible to monitor a large number of micro-climates concurrently. We have developed a system in which devices communicate their CPU measurements to an on-farm edge cloud. The edge cloud uses a combination of calibration, smoothing (noise removal), and linear regression to make predictions of the outdoor temperature at each device. We evaluate the accuracy of this approach for different temperature sensors, devices, and locations, as well as different training and calibration durations.

Author Keywords

IoT, precision agriculture, regression, sensing, prediction

INTRODUCTION

The Internet of Things (IoT) is a vision of a digitally enhanced society in which ordinary physical objects, equipped with ubiquitous network connectivity, sensors, and actuators, interact with local (edge) and remote (cloud) computing systems, to provide automation and data-driven decision support for a variety of applications. One such application where this vision is attainable in the near term is precision agriculture in which IoT technologies automate farm operations and facilitate specialized (site-specific) management of crops and livestock. The goal of precision agriculture is to reduce cost, farm labor, and resource use while increasing yields sustainability by precisely tailoring farm operations to hyper-local conditions (e.g. individual blocks or plants). Examples include automatic irrigation to maximize crop production per unit of water consumed or to synchronize harvest times, autonomous and vision-based weeding, planting, and harvesting systems, and data-driven application of inputs (fertilizers and pesticides) based on the needs of individual plants and localized soil health.

Many, if not most, IoT systems for precision agriculture depend on and integrate measurements of real time, atmospheric temperature. Temperature is used to inform and actuate irrigation scheduling, frost damage mitigation, greenhouse management, plant growth modulation, yield estimation, post-harvest monitoring, crop selection, and disease and pest management, among other farm operations [8, 24, 13, 23, 10]. Measuring and predicting temperature accurately is challenging due to variation across farm micro-climates where local temperature can deviate from the surrounding area which is typically measured at mesoscale. Measuring temperature for large number of micro-climates on a farm can be prohibitively expensive with extant weather stations and sensors (which can be hundreds to thousands of dollars).

We explore alternative ways of measuring outdoor temperature on farms by using simple, inexpensive single board computers (e.g. those in the Raspberry Pi family [22] or microcontrollers such as those in the Arduino family [4]). Our approach estimates outdoor temperature from the on-board processor temperature sensor that these devices support and which is available via their respective hardware/software interfaces. Such devices cost around $5, are battery or solar powered, and can be packaged in small, inexpensive, weatherproof enclosures, making it possible to use them in moderate and large scale geographic deployments.

To investigate how well the processor temperature of these devices can be used to predict outdoor temperature, we have developed an on-farm IoT system in which we place single-board computers *in-situ* throughout the farm. The devices transmit measurements of CPU temperature wirelessly to wall-powered, indoor, edge cloud systems [6]. We first calibrate the device CPU temperature against a co-located, high-quality temperature sensor using linear regression. We then remove the temperature sensor at each remote location.

The edge cloud computes a prediction of outdoor temperature for each device/location for each CPU measurement that it receives from the device. It does so by applying the regression coefficients from the calibration period to the CPU temperature measurement. To account for autocorrelation in the time series, we investigate the use of Single Spectrum Analysis (SSA) [9] to extract a smooth "signal" from the data prior to performing linear regression and compare this approach to non-smoothing. We also evaluate the impact of using different amounts of training data (period over which re-

gression is performed) and calibration durations. Finally, we integrate different outdoor temperature sources that include device-attached sensors (e.g. thermistors), high-end, on-farm weather stations, and remote WeatherUnderground [26] stations.

Our empirical evaluation considers two configurations. The first is a "limit study" in which we continuously update the regression coefficients using a co-located temperature sensor, to compute a one step ahead (5 minute) prediction. This configuration represents an upper bound on the efficacy of predicting outdoor temperature from processor temperature. Using a second configuration, we consider a practical application of our approach in which the edge cloud estimates the outdoor temperature (at the device) using information from the initial calibration period and the CPU temperature measurements reported by the device every 5 minutes. Our results indicate that this approach, which can reduce instrumentation costs by 50% or more, yields an average error of approximately 1.5 degrees Fahrenheit (or lower) in the farm deployments we describe. Moreover, our we show that our approach achieves similar accuracy for the different device types, locations, and ground truth weather stations/sensors that we consider.

APPROACH

We investigate the relationship between processor temperature (henceforth simply referred to as CPU temperature) embedded in single-board computers, and the atmospheric temperature that surrounds them. Our goal is to place these computers *in-situ* in agricultural settings for use as thermometers. By doing so, we can leverage their measurements to actuate and control a wide range of IoT-based farm operations, while driving down the cost of implementing such solutions at scale.

Examples of such farm operations include irrigation scheduling and frost damage mitigation strategies. For automatic irrigation scheduling, real-time temperature measurements are used to compute localized estimates of evapotranspiration (ET), which indicates the amount of water that has been lost (since the last irrigation) and that must be replaced via irrigation. Both under and over watering can decrease productivity, destroy crops, and degrade soil health. Irrigation scheduling is the most common form of IoT and data-driven decision support system on farms and is especially important for managing farms in drought stricken regions.

The terms "frost" or "freeze" are used by the public to describe a meteorological event that causes freezing injury to crops and other plants, when the air temperature falls below the tolerance level of the specific plant [17]. The ability to predict the onset of frost, its duration, and the specific locations where frost will occur is of tremendous value to the agricultural industry. In the USA, there are more economic losses to frost damage than to any other weather-related phenomenon [27]. Active frost protection strategies include application of water, use of wind engine-driven machines and heaters, and/or some combination of these methods, all of which are extremely labor intensive and costly for growers. If the onset or duration of frost is mis-predicted, the cost of any mitigation strategies applied is lost. Alternatively, incorrectly predicting that a freeze will not occur to save these costs can devastate a crop. For this reason, current practice is conservative, passing any unnecessary mitigation costs on to the consumer in exchange for a low risk of crop loss.

In both operations, accurately measuring and predicting temperature in real-time is required. However, temperature is not uniform and can vary widely across a farm, requiring that operations account for very localized differences to obtain measurable outcomes. Micro-climates can occur in large numbers due to topographic differences, surrounding structures, ground cover, plant maturity, and nearby bodies of water. Measuring temperature across vast numbers of micro-climates is costly and labor intensive given the price of high quality sensors and complexity of sensor management (data extraction, advanced analytics, connection inferences, and prediction). Many IoT vendors provide managed services to reduce this complexity for growers, but these services are expensive, require that data be transmitted off-farm to cloud based applications via cellular, and impose a recurring subscription fee on farmers in order to view their data. As a result, IoT advances have not achieved wide spread uptake in the agriculture, despite their potential.

As part of the UCSB SmartFarm effort, we are investigating ways of reducing cost and complexity of temperature-based IoT solutions, while maintaining accuracy and robustness. SmartFarm implements a low cost, on-farm edge cloud comprised of multiple Intel Next Unit of Computation (NUC) machines [12]. Using open source cloud software (AppScale [14] and Eucalyptus [20]), we design the edge clouds to be self-managing and to perform a wide range of data analytics on farm data, thereby precluding the need to transmit data off-farm and keeping cost, complexity, and latency low [15].

We use SmartFarm and single-board computers to provide accurate, real time estimates of micro-climate temperature across a farm. To do so, we place battery or solar powered devices *in-situ* in various settings and configurations within inexpensive enclosures. The devices transmit their CPU temperature wirelessly (via 802.11 or Zigbee) to an on-farm edge cloud every 5 minutes. As ground-truth, we consider co-located (device-attached) DHT digital sensors (thermistors [2]), high-end, on-farm weather stations, and WeatherUnderground remote weather service [26], which farmers commonly use to estimate temperature.

Figure 1 shows a two week time series trace (starting May 10^{th}, 2018) of CPU temperature (Pi Zero CPU) from a Raspberry Pi Zero, the outdoor temperature from an attached digital DHT22 temperature sensor (DHT Temp), and the outdoor temperature from a nearby WeatherUnderground (WU) station (WU Temp). WU measures outdoor temperature at 10 meters and the Pi Zero is at a 1 meter altitude. The Pi Zero is in a plastic enclosure with a small, covered hole from which the DHT wires exit; the DHT sensor is outdoors and hanging freely. The device is located outdoors under constant shade in Goleta, CA. We refer to this device as Pi1 in later sections of the paper. The average CPU temperature on the Pi Zero during this period is 99.71 °F with a standard deviation of 4.69.

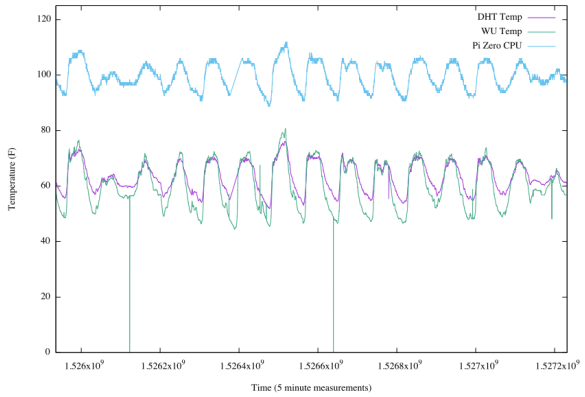


Figure 1: Two week time series trace of outdoor (device-attached DHT sensor and a nearby (WU) station) and 5-minute CPU temperature data in Fahrenheit from a Pi Zero single board computer (Pi1 in the Results section)

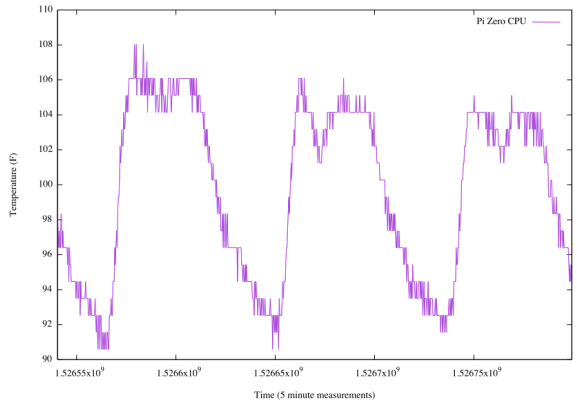


Figure 2: Two day time series sub-trace from Fig. 1 of 5-minute Pi Zero CPU temperature ($^{\circ}F$)

The mean and standard deviation for the DHT sensor and WU station are 61.93 (5.79) and 60.20 (8.35), respectively. DHT and WU temperature is similar but WU exhibits data dropout (0 values), more variance, and more extreme temperatures.

From this graph, there appears to be a correlation between CPU temperature and both outdoor temperature measures for this location. The CPU values exhibit small oscillations or noise (making the curve appear darker). A sub-portion (2 days starting May 17^{th} at midnight) of the CPU data alone is shown in Figure 2 using a different scale. We note that there are some discrepancies in the shape of different curves. We observe similar relationships using other types of devices, locations, and sources for ground-truth (e.g. DHT or WU) temperature measurements. We next investigate how accurately we can predict outdoor temperature (of these different sources) using CPU temperature of these devices.

Predicting Air Temperature from CPU Temperature

The data in Figure 1 is typical of the outdoor SmartFarm installations we have deployed suggesting that linear regression would be an effective way to predict outdoor temperature from CPU temperature. Because each single-board computer is running a multi-user operating system (Linux in this study), however, the CPU temperature exhibits fluctuations that we do not observe in the outdoor temperature. Further, because these fluctuations are caused by programs that are running on the computer, they are autocorrelated in time.

To account for this autocorrelated "noise" in the CPU temperature series, we apply Single Spectrum Analysis (SSA) [9] to the CPU series before performing regression. SSA decomposes an autocorrelated time series into "basis time series" which are analogous to principle components [1]. By summing the most significant basis series (based on a clustering of the series by eigenvalues), SSA can extract a smooth "signal" from a noisy time series. To do so, SSA requires the number of lags over which autocorrelation is significant to be supplied as a parameter.

To investigate the accuracy with which it is possible to predict outdoor temperature, our system runs multiple smoothing passes, each with a successively larger number of lags up to 12 (1 hour). During daylight and nighttime hours, outdoor temperature can be autocorrelated for several hours, but during the early morning (diurnal heating) or early evening (diurnal cooling) the significant autocorrelation duration is significantly less. For each lag we compute the coefficient of determination (R^2) for a regression covering a previous window of time and choose the number of lags that generates the highest R^2 value. We refer to this window as the *training window (TW)*. Typically (but not always) the best R^2 value is for 6 lags indicating that the significant autocorrelation in the CPU temperature series covers about 30 minutes.

The method recomputes both the smoothed series and the regression coefficients every time a new outdoor measurement is generated (every 5 minutes in this study). Thus the approach is a "piecewise" linear regression approach where the data is re-smoothed using the "best" number of lags (based on R^2 value) before each regression.

When a new CPU value arrives, we use the regression coefficients to compute a prediction of outdoor temperature. Prior to applying smoothed regression coefficients, we append the new CPU value to the training window (and remove its head, effectively sliding the window right). We compute the prediction using the smoothed CPU value (last value of the smoothed training window). We then compare this value to the actual outdoor measurement to compute the absolute difference and square difference as the error.

To summarize, the steps of our algorithm are as follows.

1. Match the temperature and CPU series using the nearest timestamps

2. Divide the matched series into a training window (TW) and test window (TE)

3. If SSA is used, smooth the CPU series using different smoothing parameters.

4. Compute the regression coefficients, i.e. y-intercept and slope, for each to model the linear relationship between

temperature (the dependent variable) and CPU (the explanatory variable) in TW

5. Extract the best parameterization for each smoothing technique using largest coefficient of determination (R^2)
6. For each matched pair of measurements in the TE, append the pair of measurements to TW and remove the first pair in TW, effectively sliding the training window right
7. Predict outdoor temperature by applying the regression coefficients to the latest CPU value (smoothed or non-smoothed), and compute and record the error (difference from actual, matched outdoor temperature); for the smoothed case, we smooth across the updated TW.
8. Repeat starting at Step 3 above and end when there are no more new measurement pairs in the test window (TE)

We refer to this configuration as a "limit study" because we believe that it provides us with an upper bound on the efficacy of our approach. However, it requires that the device and temperature sensor be co-located so that we can continuously update the regression coefficients.

We therefore consider a second configuration that does not continuously update the coefficients using the most recent temperature data. We refer this configuration as a "practical application" of our approach. For this configuration, we co-locate a temperature sensor with each device for a short, fixed period of time, which we refer to as the *calibration period*. We then remove the temperature sensor (and use it to calibrate other *in-situ* devices as needed). We apply the regression coefficients from the calibration period (which do not change) to CPU measurements reported by the device to predict the outdoor temperature at the device.

For the calibrated results, we use the algorithm above with minor modifications. The remote device transmits only its CPU measurement values via low-power radio to the edge cloud every 5 minutes. The edge cloud keeps a CPU history from the device for the same duration as the calibration period. It smooths these values if necessary and chooses the best-performing smoothing parameterization (e.g. lags) using R^2. The edge cloud then computes a prediction using the last CPU value it received (smoothed or non-smoothed). For the results in this paper, we compare this prediction against that from a co-located temperature sensor. However, we only use data from this co-located sensor to compute the prediction error after the devices have been "separated".

RESULTS

Devices and Data Sets

The devices we consider as temperature sensors in this study include the Raspberry Pi Zero Version 1.3 with a 1GHz ARMv7 processor and 512MB of RAM and the Raspberry Pi 3 Model B with a 1.2GHz ARMv8 processor and 1GB of RAM. Each Pi is equipped with 32GB of storage. We also evaluate an Arduino Uno with a ATmega328P processor with 2KB of data memory and 32KB of program memory, and an Intel Next Unit of Computation (NUC) with 8 Intel Core i7 processors (each 2.6GHz), 32GB of memory, and 1TB of SSD storage. The devices cost $5, $35, $22, and $1619 for the Pi Zero, Pi3, Uno, and NUC, respectively.

The Pi devices read their CPU temperature via a "thermal zone" which reports temperature in Celsius. The Uno reads the internal analog to digital converter using the 8th channel of the micro-controller (currently without the noise reduction feature). The NUC reads its CPU via the `sensors` utility. We convert all values to degrees Fahrenheit for this study.

The locations include a residential backyard in Goleta, CA, an experimental citrus farm at the Lindcove Research and Extension Center (LREC) in Exeter, CA, and an experimental almond farm on the campus of the California State University, in Fresno, CA. There are multiple Pi Zero devices at the Goleta location (referred to as Pi1, Pi2, Pi4, and Arduino, prefixed with "Goleta-" in the results section), a Pi Zero (LREC-PiZ) and Pi 3 (LREC-Pi3) at LREC, and a NUC at Fresno State (Fresno-NUC). All devices are in shaded, weather proof enclosures outdoors; the NUC is in a tin shed housing a powered irrigation pump next to the almond orchard. Each location is very different in terms of its vegetation and topography. LREC is located in the foot hills of the Sierra mountains; the Fresno State farm is flat and in the central valley of California; and the Goleta residence is near the ocean.

We measure atmospheric temperature (ground truth measurements used for calibration and empirical evaluation of accuracy) using device-attached temperature (AM2302 DHT22 [2]) sensors which we refer to as DHT for Goleta devices, a high end weather station at LREC called the Flux tower, and the nearest WeatherUnderground (WU) station (station 30) in Fresno. We also consider a WeatherUnderground station (station 8) for Goleta devices.

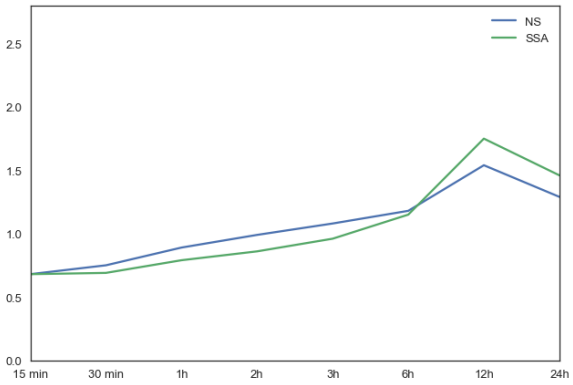
Regression, Prediction Error, and Training Window

We begin by examining the effect of smoothing on each regression as part of a "limit study". To do so, we compare SSA and no smoothing over a number of different training window sizes. As described previously, we use the regression coefficients for the number of lags for SSA that results in the highest R^2 value. We detail the effect of using smoothing and training window size to enhance regression on temperature *prediction*. At time step t we predict the outdoor temperature at time step $t + 1$ (5 minutes later). Since an application may need the temperature at an arbitrary moment in time (and not on a precise 5-minute periodicity), this prediction error serves as an upper bound on the error that an application which is not time synchronized with the measurement system might experience. We then compare different sources for predictions (locally attached DHT vs Internet-accessible WeatherUnderground) and we conclude with results showing the application of our approach in a practical IoT setting.

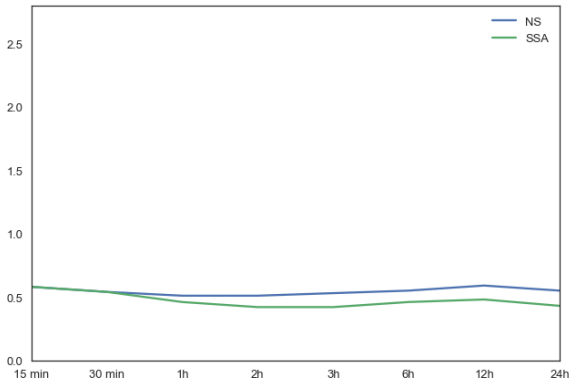
Prediction Error

In Figure 3 we show the Mean Absolute Error (MAE) for the one-step ahead prediction as a function of history size [1]. Each graph compares the effect on prediction accuracy of different

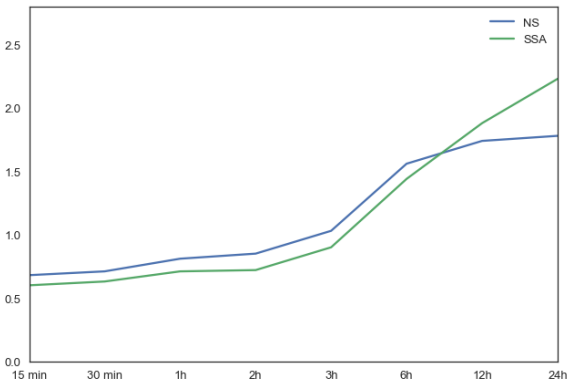
[1]Most typically, the prediction error is presented as the Mean Square Error (MSE) or the Root Mean Square Error (RMSE) in an error analysis. While these statistics offer insights into the distributional



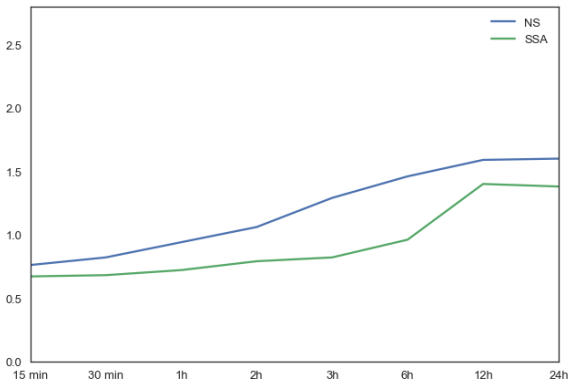
(a) MAE for Goleta-Pi1-DHT



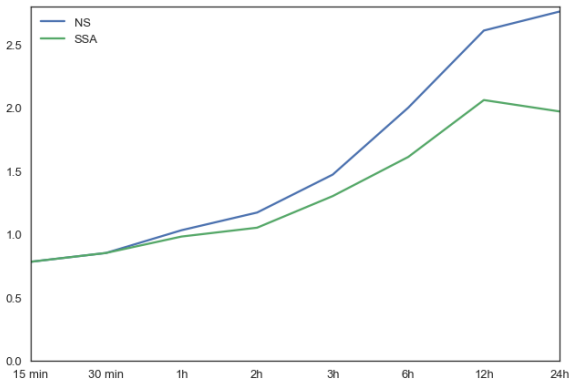
(b) MAE for Goleta-Arduino-DHT



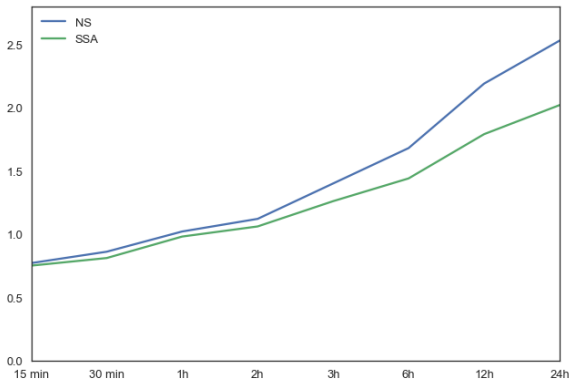
(c) MAE for Goleta-Pi1-WU



(d) MAE for Fresno-NUC-WU



(e) MAE for LREC-Pi3



(f) MAE for LREC-PiZ

Figure 3: Mean Absolute Error in degrees Fahrenheit for predictions of outdoor from CPU temperature of different devices, locations, and sources of ground-truth temperature (DHT= high quality temperature sensor; WU=WeatherUnderground; LREC=high-end on-farm weather station) for two methods: Non-Smoothing (NS) and Single Spectrum Analysis (SSA).

	Pi1		Pi2		Pi4	
h	NS	SSA	NS	SSA	NS	SSA
1	5.5	4.5	6.4	6.5	10.4	14.6
4	5.5	4.5	2.6	2.4	1.9	1.6
8	2.7	2.6	1.5	1.5	1.4	1.4
12	2.2	2.2	1.5	1.6	1.5	1.6
24	1.4	1.4	1.4	1.4	1.5	1.5
48	1.5	1.6	1.4	1.4	1.3	1.3
72	1.3	1.3	1.4	1.4	1.3	1.4
96	1.3	1.4	1.4	1.4	1.3	1.4
168	4.8	4.8	1.4	1.4	1.3	1.3
336	4.8	4.8	1.4	1.4	1.3	1.3

Table 1: Mean Absolute Error ($^\circ F$) with No Smoothing (NS) and SSA for different calibration periods in hours (h).

smoothing methods for the different locations and devices for a prediction period of 3 days. The x-axis is the training window size; the y-axis shows errors in $^\circ F$.

From the graphs in Figure 3, we see that SSA improves prediction accuracy compared to the absence of smoothing (NS). In this study, 15 minutes corresponds to 3 measurements. When the temperature is slowly changing (e.g. the CPU temperature does not change over a 15 minute period) regression becomes numerically unstable (i.e. the co-variance matrix has values that are nearly zero on the diagonal). Compared to Fresno or Goleta, for example, the CPU temperatures at LREC is more stable since the devices are sited near a large irrigation reservoir. SSA smooths the previous 3 measurements more than the other methods, occasionally generating regression coefficients that are very large or numerically infinite (i.e. NaN) as a result of trying to invert the co-variance matrix. Our system detects this condition and disables smoothing when it leads to a failed regression.

Also note that the errors are relatively small. All of the locations we have tested are located in California and during the prediction periods, the temperature varied from the mid 40s to the mid 80s $^\circ F$. In each case, the MAE error is under $1^\circ F$ for a TW of 1 hour or less. The Arduino Uno (Goleta-Arduino-DHT) produces the lowest error and the error does not grow with window size. We believe this is due to the very consistent and slowly changing temperature of the location during the prediction period (i.e. it is nearer to the ocean than the other Goleta devices and the 3 day prediction period is in May vs March for the other locations). The accuracy of our approach is similar regardless of location (e.g. Goleta, Lindcove (LREC), or Fresno) and source of ground truth temperature measurement (DHT, Flux tower (LREC), or WU) for a TW of 1 hour or less.

Practical Application

The data and analysis presented in the previous subsection show the minimum error that is possible. That is, they verify that it is possible to predict the outdoor temperature from the internal CPU temperature sensor with a high degree of accuracy in a variety of meteorological settings. To be practically useful, however, the technique must be able to predict outdoor temperature without the presence of an outdoor thermometer (i.e. from CPU temperature alone). That is, our goal is to investigate whether we can use the CPU temperature sensor (which will be present by virtue of the need for a controller) as a replacement for a localized outdoor thermometer.

Specifically, in a practical application of this technique, with no outdoor thermometer, it is not possible to perform a regression at each time step using the current outdoor temperature reading. Instead, our approach is to generate a regression coefficients from a *calibration period* that we then use over a later prediction period. We site single-board computers in each location with an attached DHT outdoor temperature sensor for a fixed, continuous calibration period. Then we remove the DHT sensor (so it can be used for another calibration) and estimate outdoor temperature from the computer's CPU temperature using the regression coefficients we computed at the end of the calibration period. Table 1 shows the Mean Absolute Errors for different calibration periods and three Pi Zero devices.

Pi1, Pi2, and Pi4 are all Raspberry Pi Zero single-board computers with externally attached DHT temperature sensors. All three were located in the same outdoor setting in Goleta, California. We chose a random date between January 1^{st}, 2018 and May 15^{th}, 2018 in each case to use as the start of the test/prediction period. We installed the Arduino too late to include in this study, but we plan to include it once we collect sufficient data. In each experiment, we use a trace of the DHT external measurements and the corresponding CPU measurements over a fixed calibration period (shown in column 1, measured in hours) to compute a set of regression coefficients. We then use the coefficients to predict the DHT measurements from the CPU measurements (without re-regressing) for the next two weeks following the calibration period. Columns 2 through 7 show the Mean Absolute Error (MAE) during the measurement period immediately following calibration without smoothing and with SSA for the calibration regression. Thus, this table shows the errors when one set of regression coefficients is used to predict the next two weeks of outdoor temperature (as a function of calibration period).

While SSA improves the errors in the piecewise regression case (cf Figure 3), it is less effective when one set of coefficients must be used over a long period of time when a sufficiently long calibration period is available. Note that for some short calibration periods, SSA can improve accuracy, but only when there is sufficient variation to maintain numerical stability in the regression. Further, the calibration period should include at least one full diurnal cycle to be effective. Finally, the minimum error is consistently $1.3^\circ F$ or $1.4^\circ F$.

Finally, not all time periods during a diurnal cycle may be needed for certain applications. As part of SmartFarm, for example, we are developing a new algorithm for computing localized evapotranspiration (ET) [21]. ET is an often-used metric for computing crop water stress or water requirements and it is typically based on meteorological measurements that cover large areas (e.g. a county or zip code). ET computa-

properties of the errors, our experience with professional agricultural personnel has led us to concentrate on the MAE as a practical error metric since it can be interpreted as how far "off" the measurements are "on the average." We omitted the RMSE results in favor of brevity.

	Pi1		Pi2		Pi4	
h	NS	SSA	NS	SSA	NS	SSA
24	1.1	1.4	2.5	2.1	1.3	1.2
48	1.2	1.6	0.7	0.9	0.9	1.3
72	1.1	1.2	0.7	1.0	1.3	0.9
96	1.1	1.2	0.8	1.0	0.9	1.0
168	4.1	4.1	0.7	0.7	0.8	0.8
336	4.1	4.1	0.7	0.7	0.8	0.8

Table 2: Mean Absolute Error in degrees Fahrenheit with No Smoothing (NS) and SSA for different calibration periods (measured in hours (h)) using data from noon to 3 PM.

	Pi1		Pi2		Pi4	
h	NS	SSA	NS	SSA	NS	SSA
24	1.1	1.4	1.4	1.4	1.3	1.4
48	1.2	1.4	1.4	1.4	1.3	1.3
72	1.1	1.2	1.5	1.4	1.3	1.3
96	1.3	1.4	1.5	1.4	1.3	1.3
168	4.1	4.1	1.5	1.5	1.4	1.4
336	4.1	4.1	1.6	1.6	1.4	1.3

Table 3: Mean Absolute Error in degrees Fahrenheit with No Smoothing (NS) and SSA for different calibration periods (measured in hours (h)) using data from 10 PM to 7 AM.

tions rely, in part, on outdoor temperature measured during "solar max" – typically between noon and 3 PM in North America. Similarly, frost prevention using wind machines mixes warm air aloft (e.g. at 10 meters) with colder air that has settled near the ground during the nighttime hours (e.g. between 10:00 PM and 7:00 AM). Thus, it may be that it is possible to obtain more accurate measurements by including only those hours that are of interest during a diurnal cycle.

Tables 2 and 3 show the MAE for non-smoothed and SSA calibration using only data gathered from noon to 3 PM and from 10 PM to 7 AM respectively. We show only results for calibration periods of at least 24 hours since the calibration period must span at least one diurnal cycle. In most cases (particularly for the solar max predictions) the best prediction (lowest MAE) improves when we use only the periods of interest for the regression. However, the improvements are small in absolute terms (often $0.1^{\circ}F$). We have yet to determine whether the additional accuracy is necessary for either localized ET calculation or frost prevention. Doing so is the subject of the on-going SmartFarm work that is leveraging this technique.

RELATED WORK

Accurate micro-climate modeling is essential for agriculture operations such as irrigation scheduling and frost protection [8, 24, 13, 23, 10]. We investigate the use of simple, low cost, single board computers to estimate air temperature for use in these applications. Although such devices are increasingly integrated into IoT solutions for agriculture (e.g. providing alerts, irrigation control, communication of sensor data [19, 15, 25, 18, 5, 30]), there are no studies of which we are aware that use the devices themselves as thermometers.

To enable this, we estimate outdoor temperature from CPU temperature linear regression of temperature time series. Others have shown that doing so is useful for other applications and analyses[11, 28, 16, 29]. Our work is complementary to these and is unique in that it combines SSA (noise reduction) with regression to improve prediction accuracy. As in other work, we leverage edge computing to facilitate low latency response and actuation for IoT systems [3, 7].

CONCLUSIONS

In this paper, we investigate an alternative, low cost way of measuring and predicting outdoor temperature using inexpensive, single board computers as temperature sensors. Our approach uses linear regression to model the relationship between outdoor temperature and device CPU temperature at each device and employs SSA to account for autocorrelation in the time series. We calibrate each in-situ device using a high quality temperature sensor for a fixed duration of time; we use the regression coefficients from the calibration period (which do not change) to predict outdoor temperature from CPU temperature thereafter, using different devices and ground truth temperature sensor/stations (e.g. on-farm weather station, thermistor, WeatherUnderground station). We empirically evaluate the approach using different amounts of training data, calibration durations, and locations. Our results show that this approach can generate average errors of $1.5^{\circ}F$ or lower in real-world precision agricultural deployments.

As part of on-going work, we continue to collect data from different sites and devices for analysis and prediction. Going forward, we plan to investigate the impact of deployment characteristics (e.g. humidity, wind speed, shade vs full sun) and device use (computation and communication) on prediction accuracy. In addition, we will investigate the use of other time series prediction techniques to perform prediction and identify other ways in which simple, single board computers can be used to infer and predict environmental conditions.

REFERENCES

1. Abdi, H., and Williams, L. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics 2*, 4 (2010).
2. Adafruit AM2302 Wired DHT22 Temperature and Humidity Sensor. [Online; accessed 22-Jun-2018] `https://www.adafruit.com/product/393`.
3. Alturki, B., Reiff-Marganiec, S., and Perera, C. A hybrid approach for data analytics for internet of things. In *Int. Conf. on the Internet of Things* (2017).
4. Arduino. [Online; accessed 15-Nov-2017] `https://www.arduino.cc`.
5. Beckwith, R., Teibel, D., and Bowen, P. Report from the field: results from an agricultural wireless sensor network. In *Local Computer Networks* (2004).
6. Elias, A. R., Golubovic, N., Krintz, C., and Wolski, R. Wheres the bear?–automating wildlife image processing using iot and edge cloud systems. In *ACM Conference on IoT Design and Implementation* (2017).
7. Feng, L., Kortoçi, P., and Liu, Y. A multi-tier data reduction mechanism for iot sensors. In *Intl Conf on the Internet of Things* (2017), 6.

8. Ghaemi, A. A., Rafiee, M. R., and Sepaskhah, A. R. Tree-temperature monitoring for frost protection of orchards in semi-arid regions using sprinkler irrigation. *Agricultural Sciences in China 8*, 1 (2009), 98–107.

9. Golyandina, N., and Zhigljavsky, A. *Singular Spectrum Analysis for time series*. Springer Science & Business Media, 2013.

10. Gonzalez-Dugoa, V., Zarco-Tejadaa, P., Bernia, J., Suareza, L., Goldhamerb, D., and Fereres, E. Almond tree canopy temperature reveals intra-crown variability that is water stress-dependent. Tech. rep., Agricultural and Forest Meteorology, 2011.

11. Guestrin, C., Bodik, P., Thibaux, R., Paskin, M., and Madden, S. Distributed regression: an efficient framework for modeling sensor network data. In *Intl Symp on Information processing in sensor networks* (2004).

12. Intel NUC. `https://en.wikipedia.org/wiki/Next_Unit_of_Computing` [Online; accessed 1-Feb-2018].

13. Ioslovich, I., Sylaios, G., Plauborg, F., and Battilani, A. Optimal model-based deficit irrigation scheduling using aquacrop: A simulation study with cotton, potato and tomato. *Agricultural Water Management 163* (2016).

14. Krintz, C. The appscale cloud platform: Enabling portable, scalable web application deployment. In *Internet Computing*, IEEE (2013).

15. Krintz, C., Wolski, R., Golubovic, N., Lampel, B., Kulkarni, V., Sethuramasamyraja, B., Roberts, B., and Liu, B. SmartFarm: Improving Agriculture Sustainability Using Modern Information Technology. In *KDD Workshop on Data Science for Food, Energy, and Water* (Aug. 2016).

16. Lane, N. D., Bhattacharya, S., Georgiev, P., Forlivesi, C., Jiao, L., Qendro, L., and Kawsar, F. Deepx: A software accelerator for low-power deep learning inference on mobile devices. In *Information Processing in Sensor Networks (IPSN)* (2016).

17. Levitt, J., et al. *Responses of Plants to Environmental Stress, Volume 1: Chilling, Freezing, and High Temperature Stresses.* Academic Press., 1980.

18. N. Golubovic and C. Krintz and R. Wolski and S. Lafia and T. Hervey and W. Kuhn. Extracting Spatial Information from Social Media in Support of Agricultural Management Decisions. In *ACM SIGSPATIAL Workshop on Geographic Information Retrieval* (2016).

19. Nikolidakis, S. A., Kandris, D., Vergados, D. D., and Douligeris, C. Energy efficient automated control of irrigation in agriculture by using wireless sensor networks. *Computers and Electronics in Agriculture 113* (2015), 154–163.

20. Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D. The eucalyptus open-source cloud-computing system. In *IEEE Cluster Computing and the Grid* (2009).

21. Penman, H. L. Natural evaporation from open water, bare soil and grass. *Proc. R. Soc. Lond. A 193*, 1032 (1948), 120–145.

22. Rasberry Pi. `https://www.raspberrypi.org`. [Online; accessed 15-Nov-2016].

23. Roberts, B., Fritschi, F., Horwath, W., and Bardhan, S. Nitrogen Mineralization potential as influenced by microbial biomass, cotton residues and temperature. *Plant Nutrition* (2013).

24. Stombaugh, T. S., Heinemann, P., Morrow, C., and Goulart, B. Automation of a pulsed irrigation system for frost protection of strawberries. *Applied Engineering in Agriculture 8*, 5 (1992), 597–602.

25. Vasisht, D., Kapetanovic, Z., Won, J., Jin, X., Chandra, R., Sinha, S. N., Kapoor, A., Sudarshan, M., and Stratman, S. Farmbeats: An iot platform for data-driven agriculture. In *NSDI* (2017), 515–529.

26. WeatherUnderground. [Online; accessed 22-Jun-2018] `http://www.weatherunderground.com/`.

27. White, G., and Haas, J. Assessment of Research on Natural Hazards. Tech. rep., MIT Press, 1975.

28. Xie, C., Tank, A., Greaves-Tunnell, A., and Fox, E. A unified framework for long range and cold start forecasting of seasonal profiles in time series. *stat 1050* (2017), 23.

29. Yao, S., Hu, S., Zhao, Y., Zhang, A., and Abdelzaher, T. Deepsense: A unified deep learning framework for time-series mobile sensing data processing. In *WWW* (2017).

30. Zheleva, M., Bogdanov, P., Zois, D.-S., Xiong, W., Chandra, R., and Kimball, M. Smallholder agriculture in the information age: Limits and opportunities. In *Workshop on Computing Within Limits* (2017).